

# Анализ полученных результатов

АиСД-2, 2023

Хорасанджян Левон Арменович, БПИ218

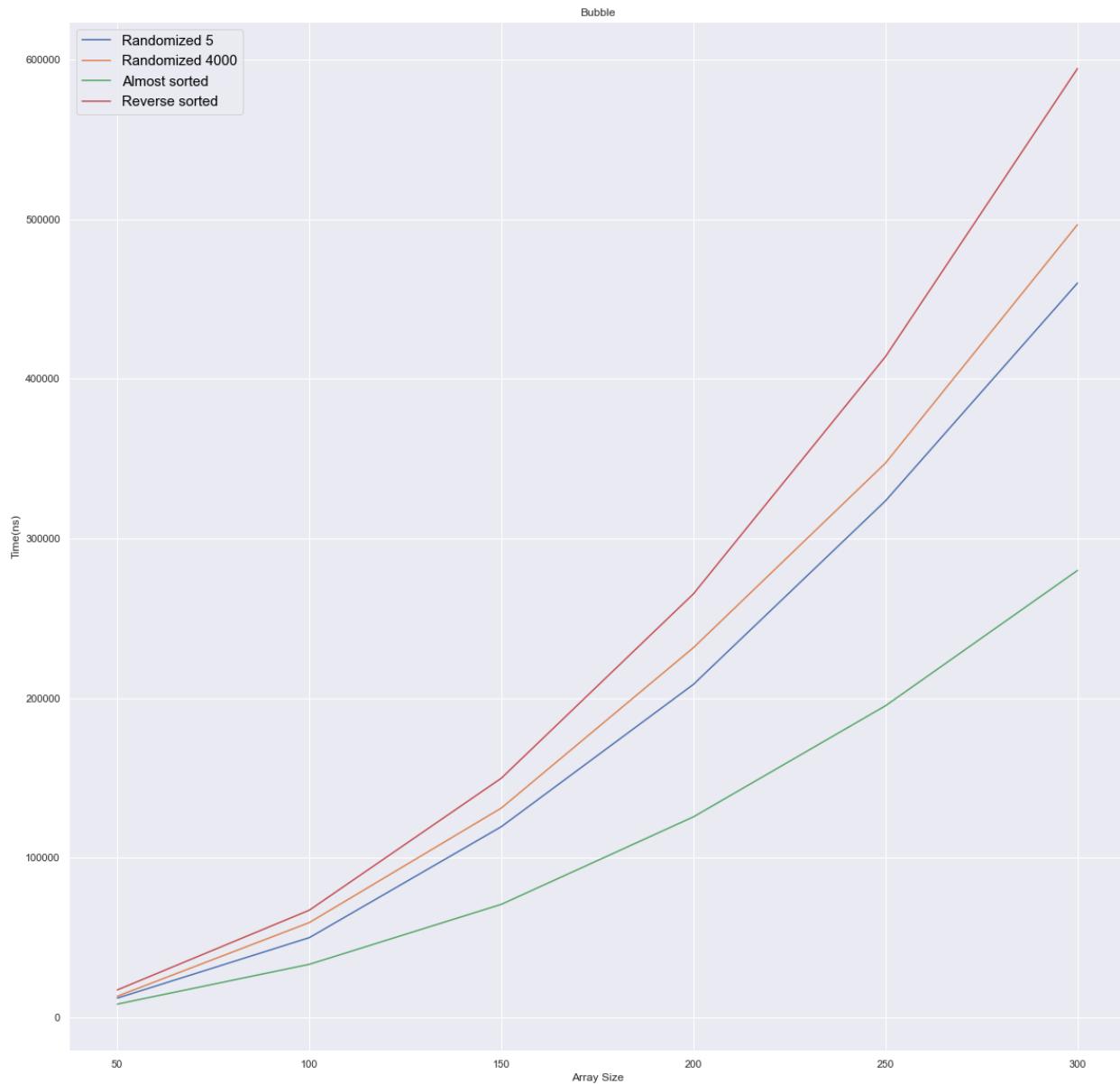
Код написан и запущен в CLion

Выполнены все задачи:

- 1) Написаны реализации сортировок;
- 2) Засечено время работы сортировок, посчитано кол-во элементарных операций, сделана проверка на корректность полученных отсортированных массивов;
- 3) Результаты записаны в .csv-файлы, после чего на их основе были сформированы графики с помощью .ipynb-файлов;
- 4) Сформированы Excel-таблицы на основе .csv-файлов.

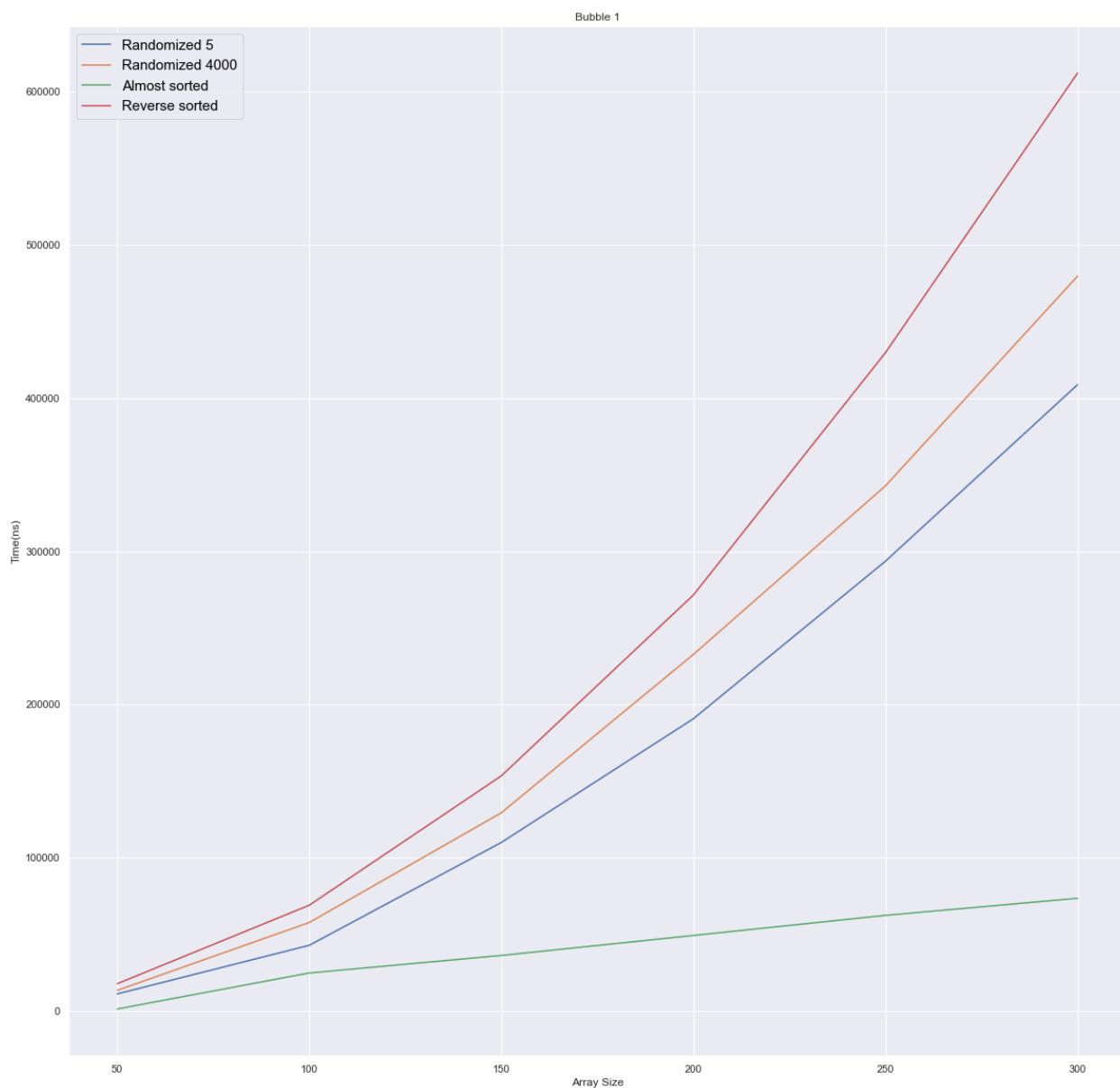
# Подсчёт времени

## Пузырьком



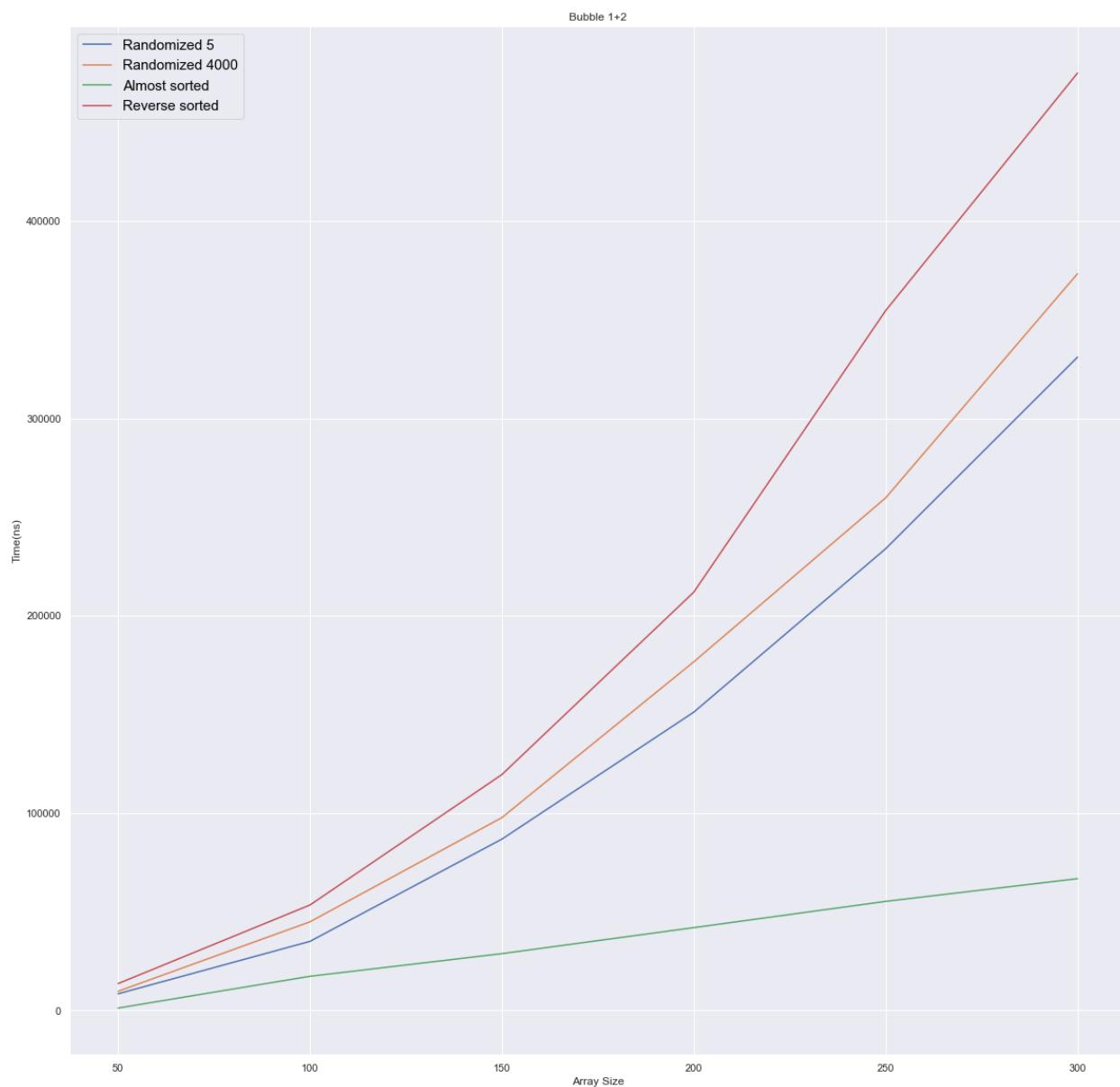
Как и ожидалось, худшим образом себя проявляет на обратно отсортированном массиве, в то время как на почти отсортированном меньше всего приходится делать перестановок.

## Пузырьком с оптимизацией Айверсона 1



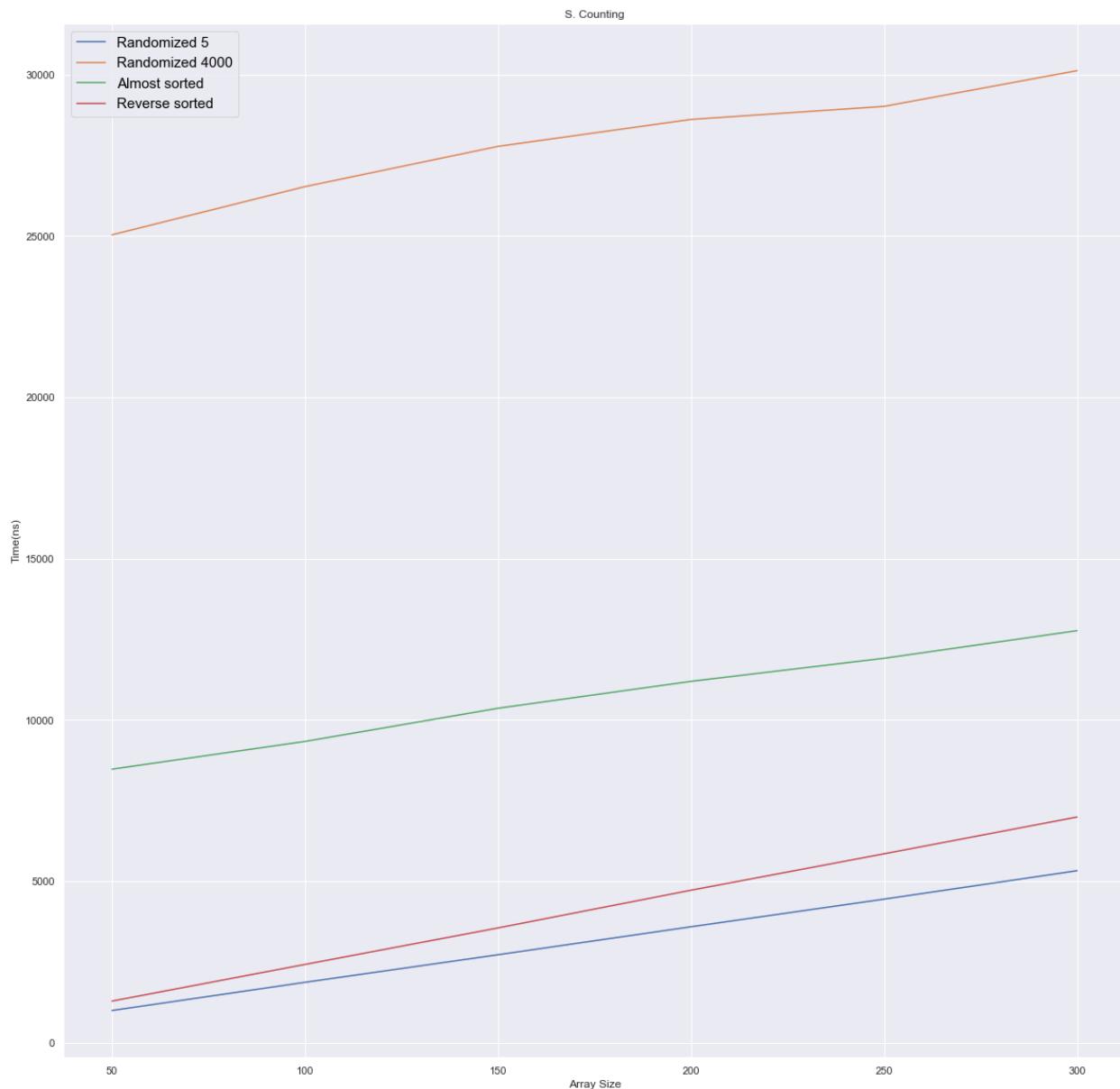
Всё аналогично предыдущему результату, однако у всех графиков время работы намного ниже, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## **Пузырьком с оптимизацией Айверсона 1+2**



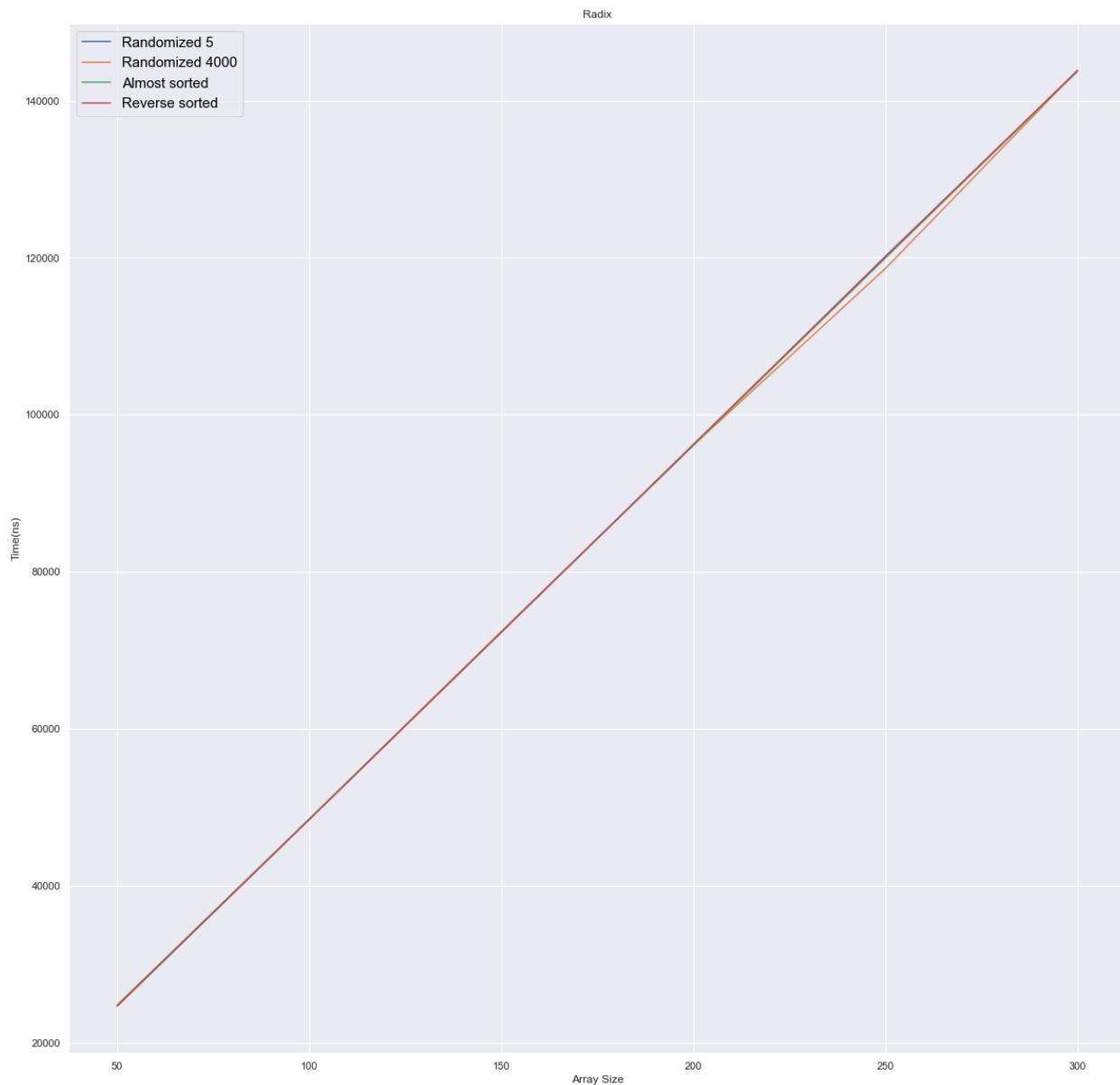
Всё аналогично предыдущему результату, однако у всех графиков время работы намного ниже, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Подсчётом



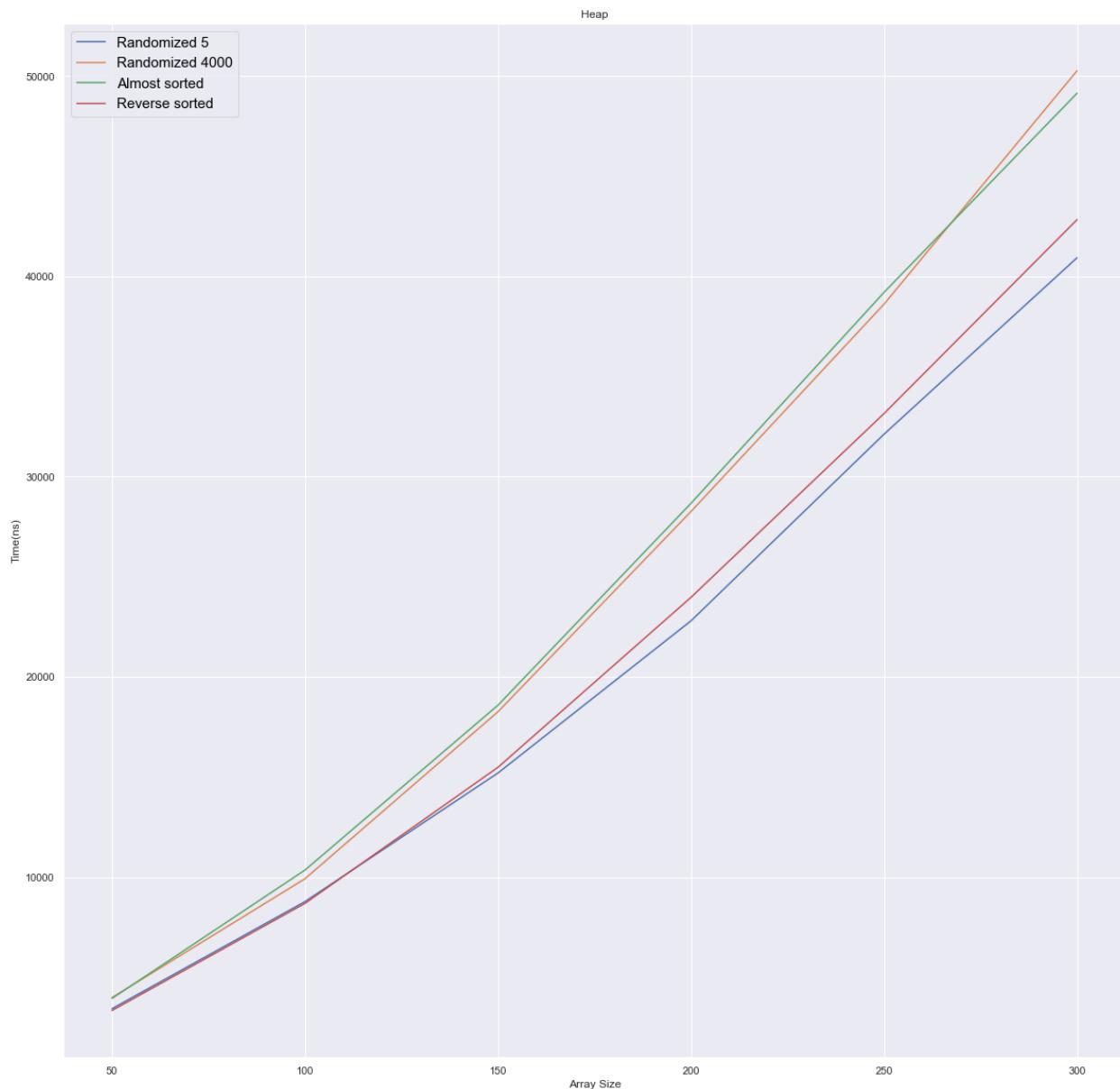
Как и ожидалось, хуже всего работает на числах с большим разбросом (модуль разности максимума и минимума), поэтому синий график самый быстрый, а жёлтый — самый медленный.

## Цифровая



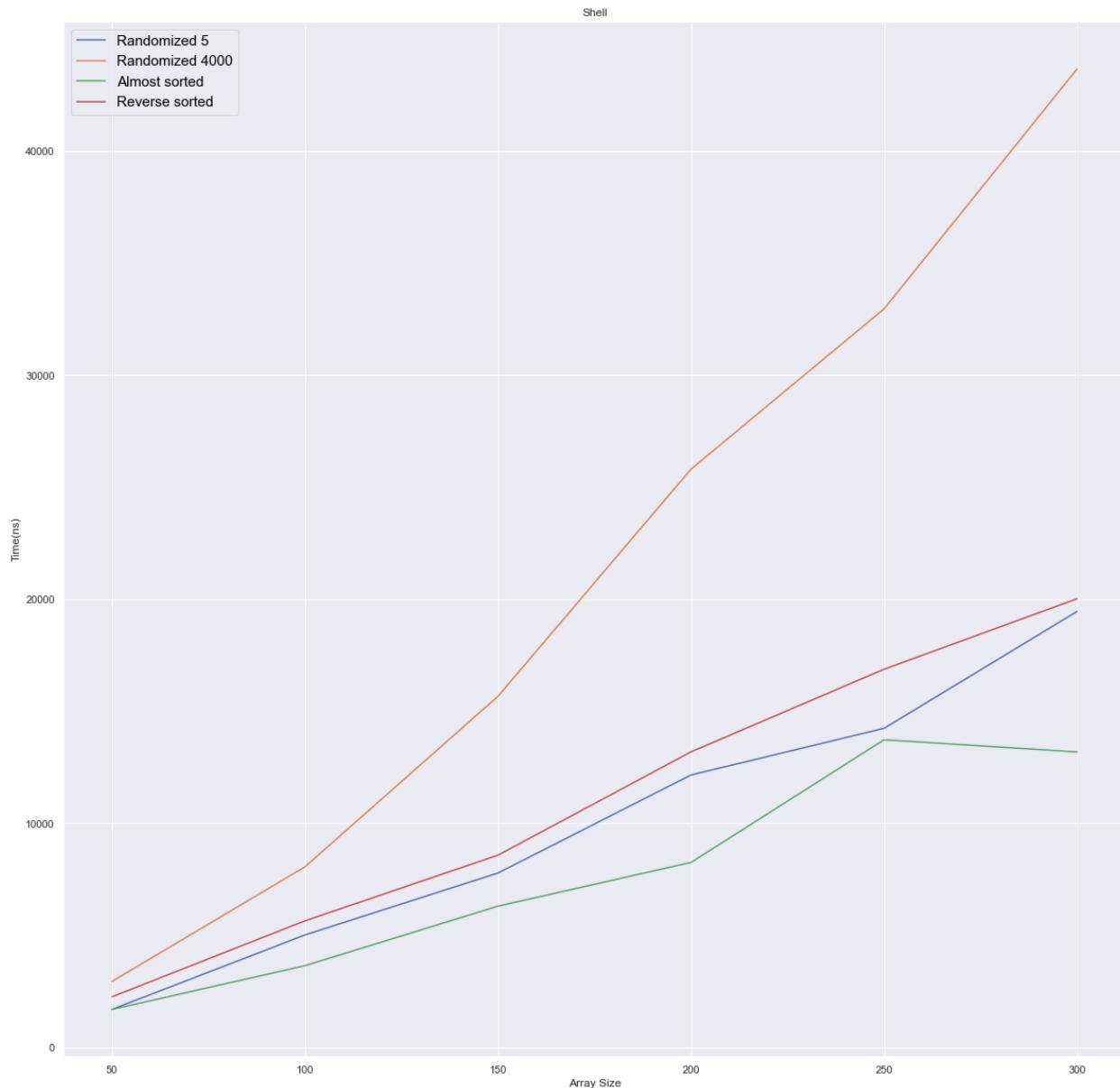
Видим, что цифровая сортировка работает линейно и одинаково для всех типов массивов.

## Пирамидальная



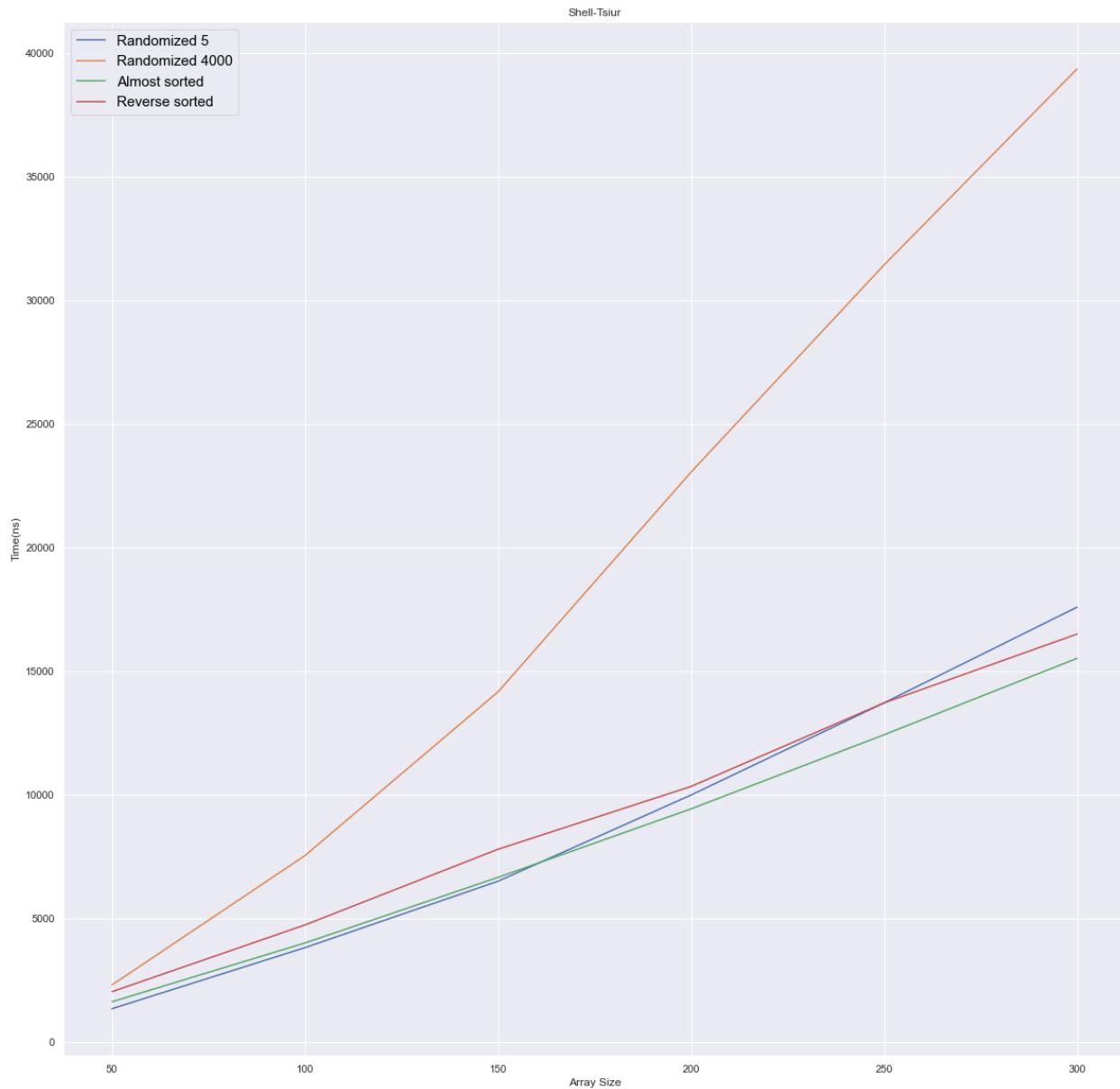
Лучше работает на обратно отсортированном массиве и с маленьким диапазоном значений.

## Шелла



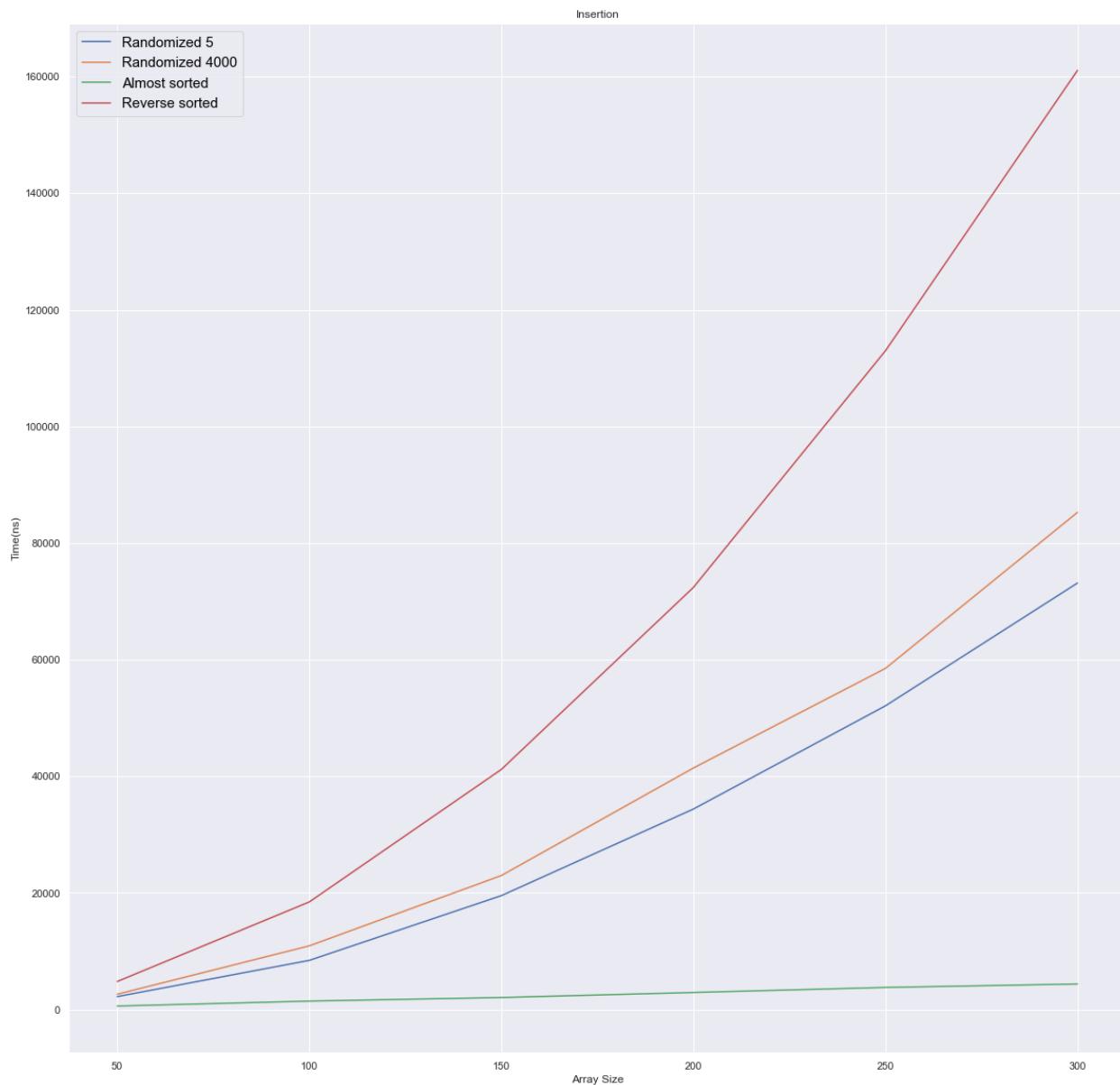
Хуже всего работает на случайном массиве с диапазоном [1;4000], поскольку слишком часто приходится производить перестановки элементов.

## Шелла-Циура



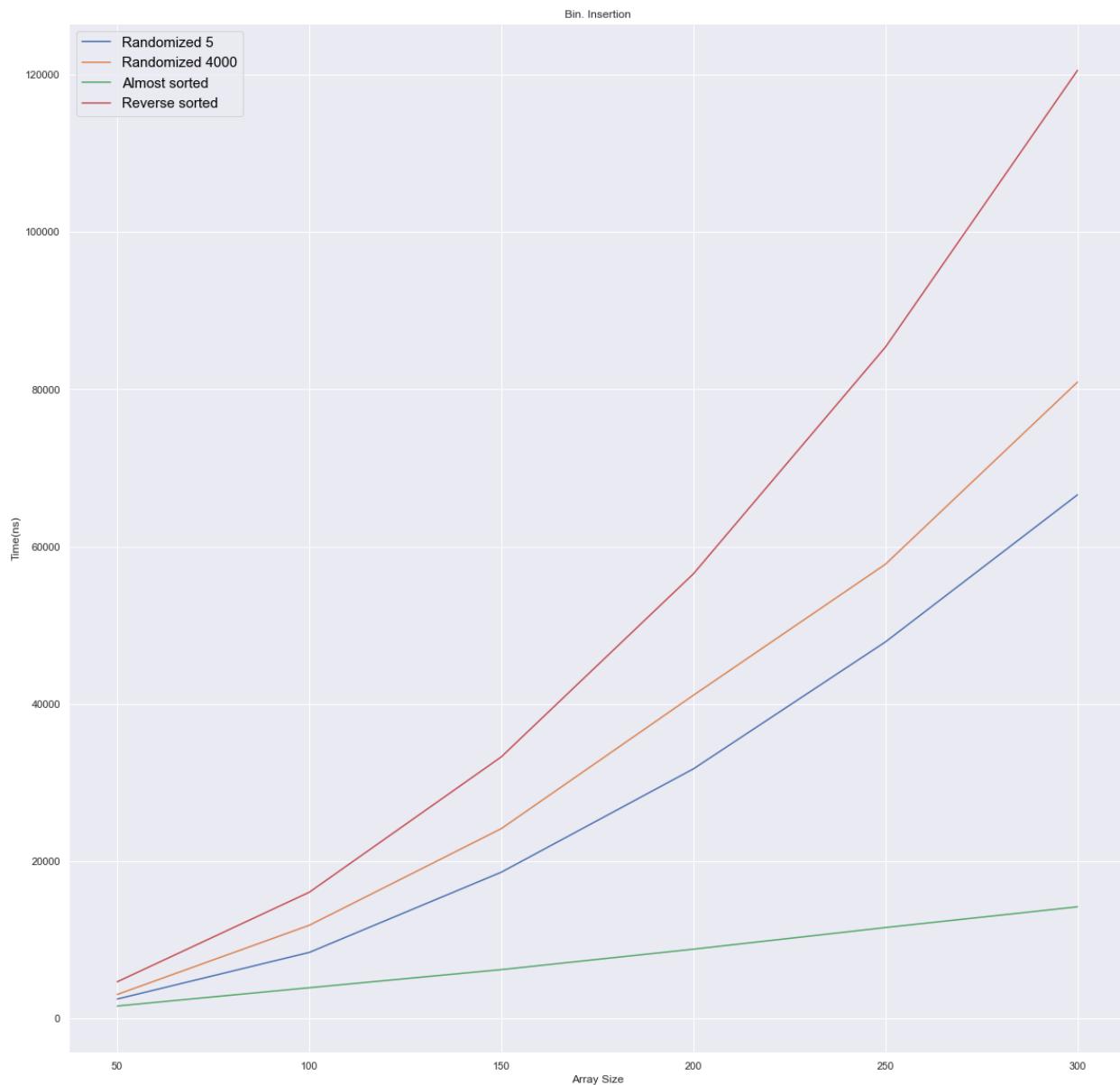
В отличии от предыдущей ситуации, здесь в целом сократилось время работы у каждого типа массива, при этом синий и зелёный графики стали равномерно идти при увеличении размера массива.

## *Вставками*



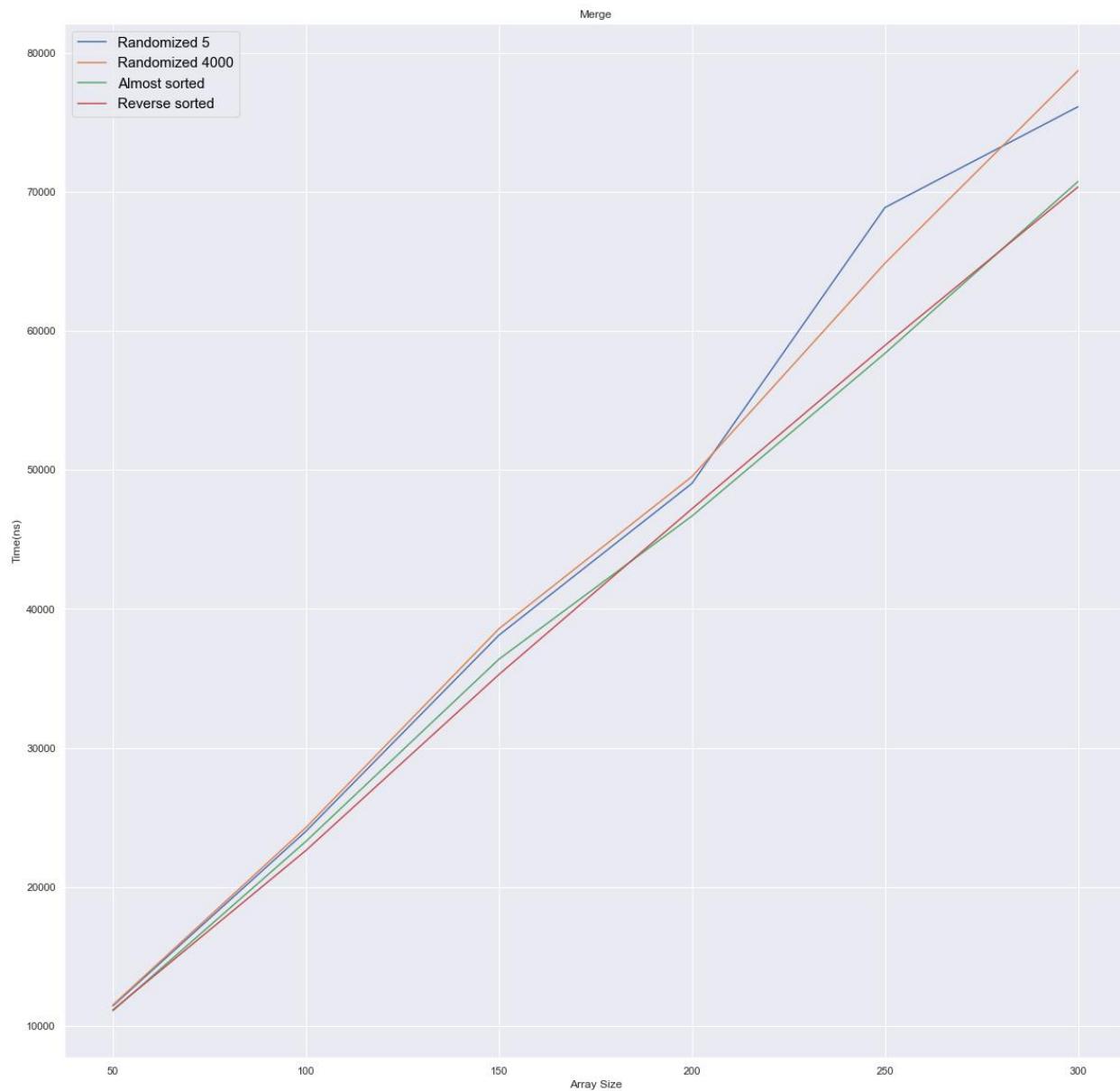
Как и ожидалось, очень медленно работает с обратно отсортированным массивом, поскольку каждый раз приходится проходить весь отрезок до начала, прежде чем вставить элемент на нужную позицию (`array[0]`).  
Лучше всего показала себя при работе с почти отсортированным, поскольку в большинстве случаев элемент не приходится вставлять в другую позицию уже отсортированного подотрезка.

## Бинарными вставками



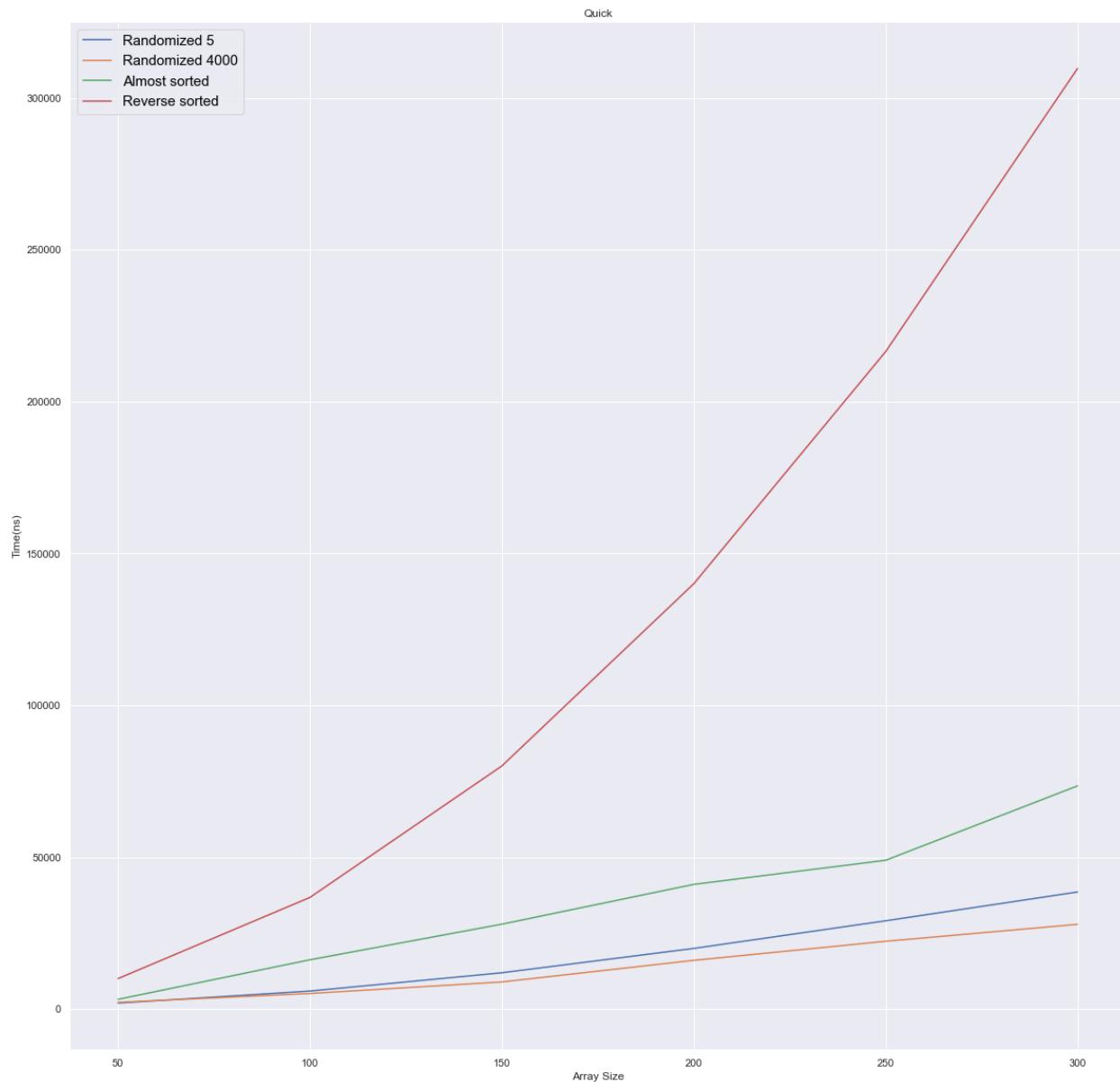
Время работы всех массивов, кроме почти отсортированного, немного сократилось, а так наблюдается ситуация, аналогичная предыдущему случаю.

## **Слиянием**



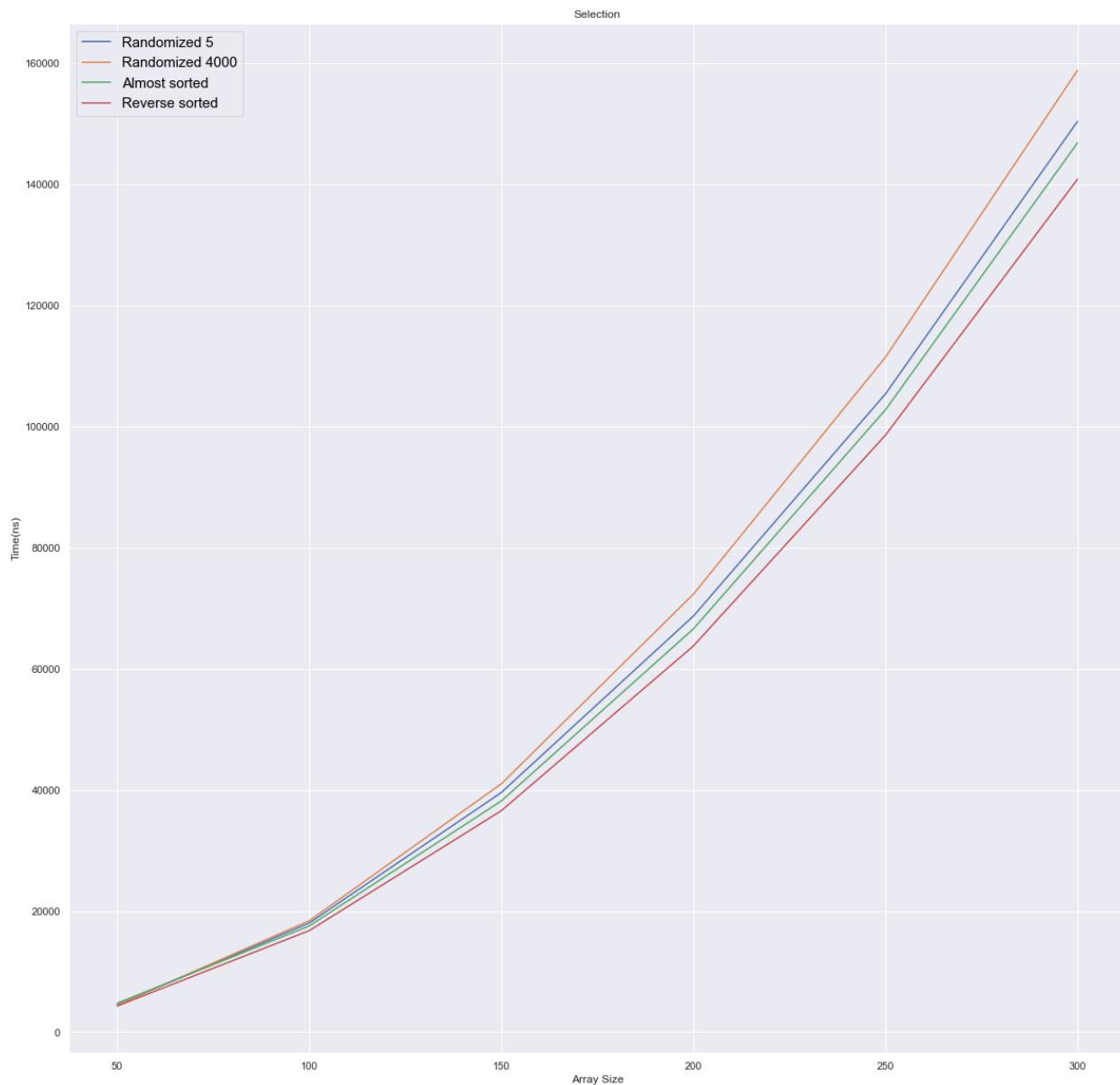
Лучше всего работает на максимально упорядоченных массивах (по возрастанию и убыванию). Графики расходятся с увеличением размера массивов.

## **Быстрая**



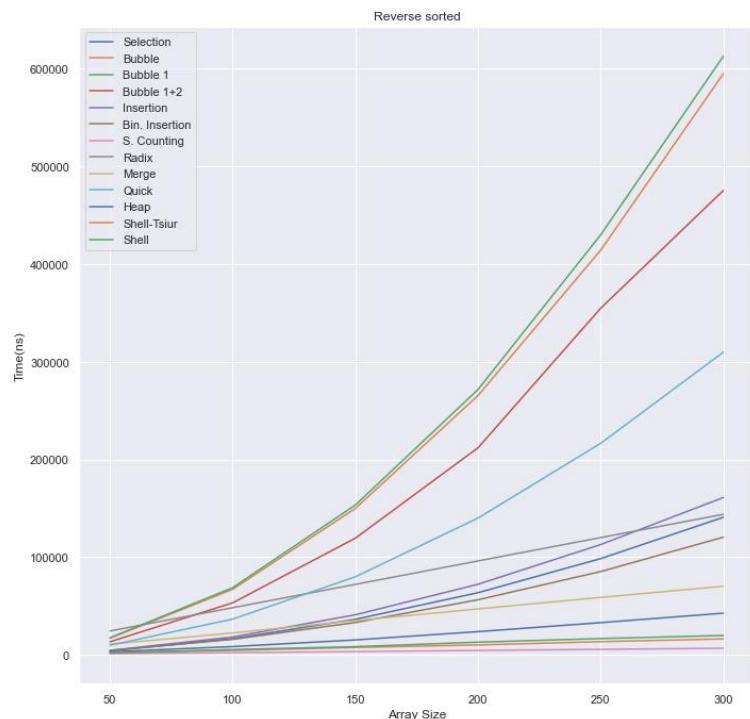
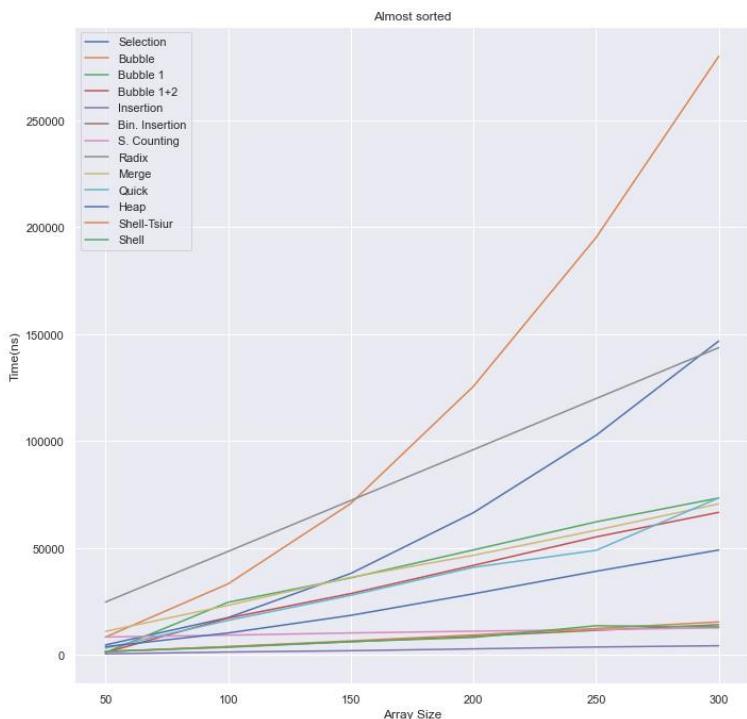
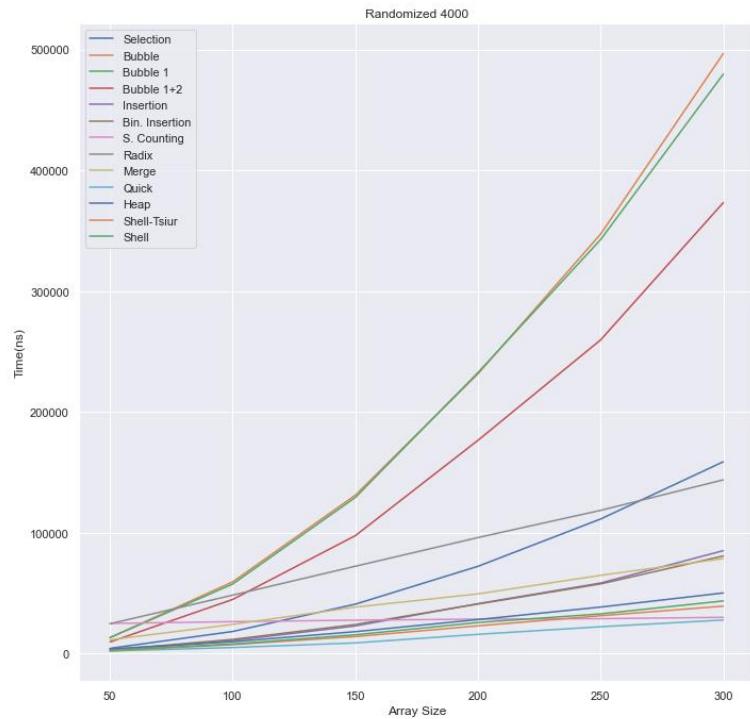
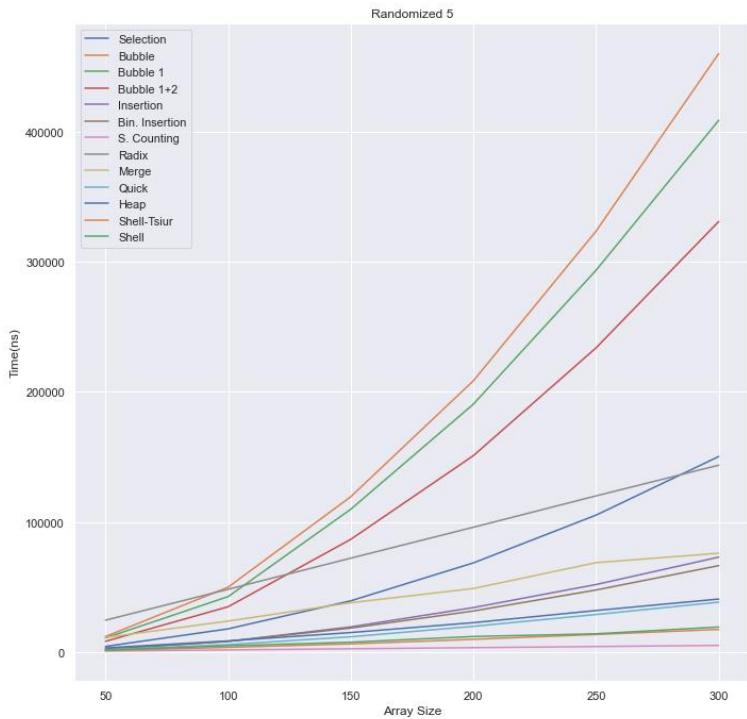
Очень медленно работает на обратно отсортированном массиве, поскольку за pivot берётся первый элемент подотрезка (т. е. максимальный для текущего подотрезка).

## **Выбором**



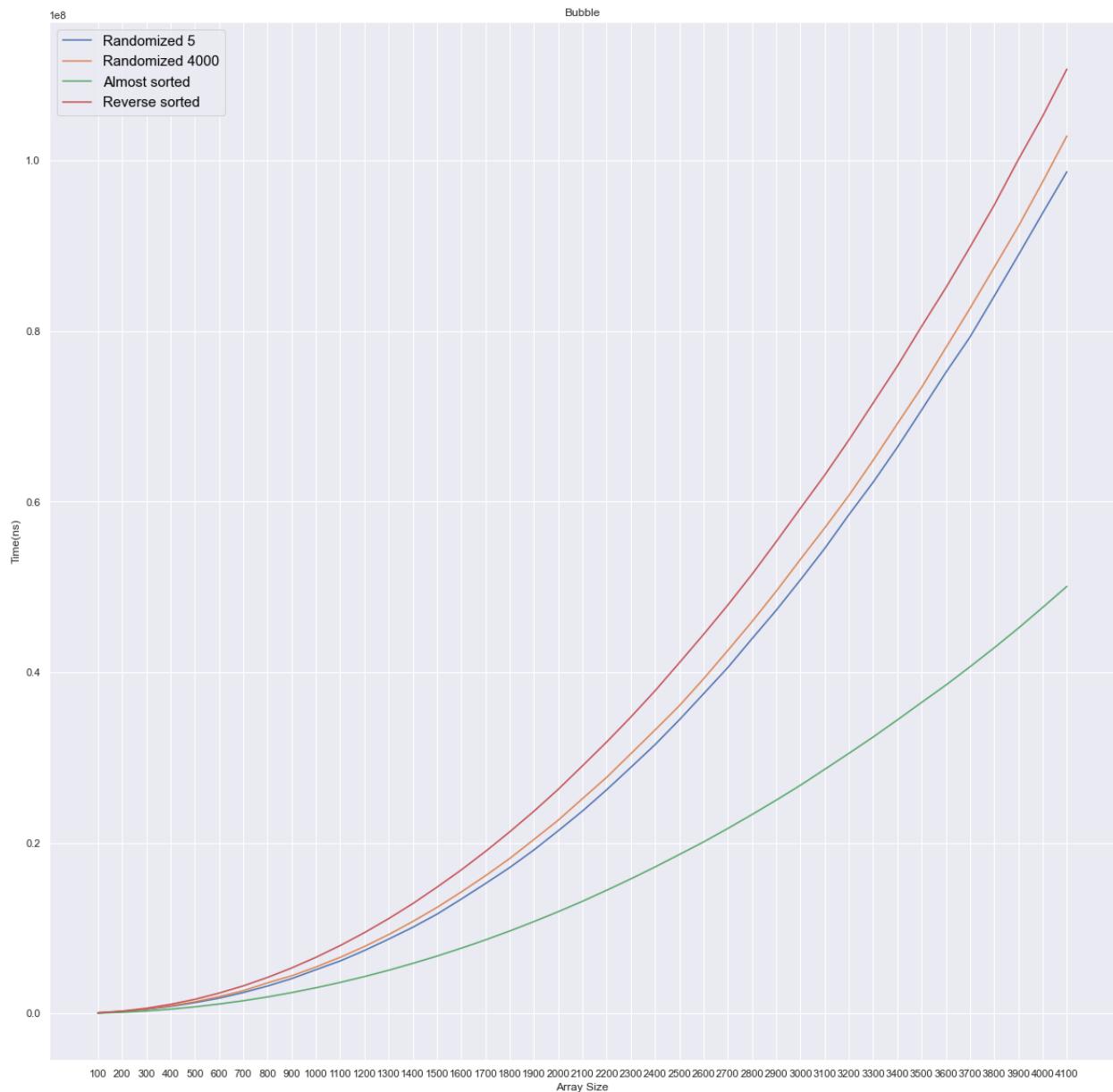
Почти одинаково работает на всём диапазоне размера массива.

# Сравнение сортировок



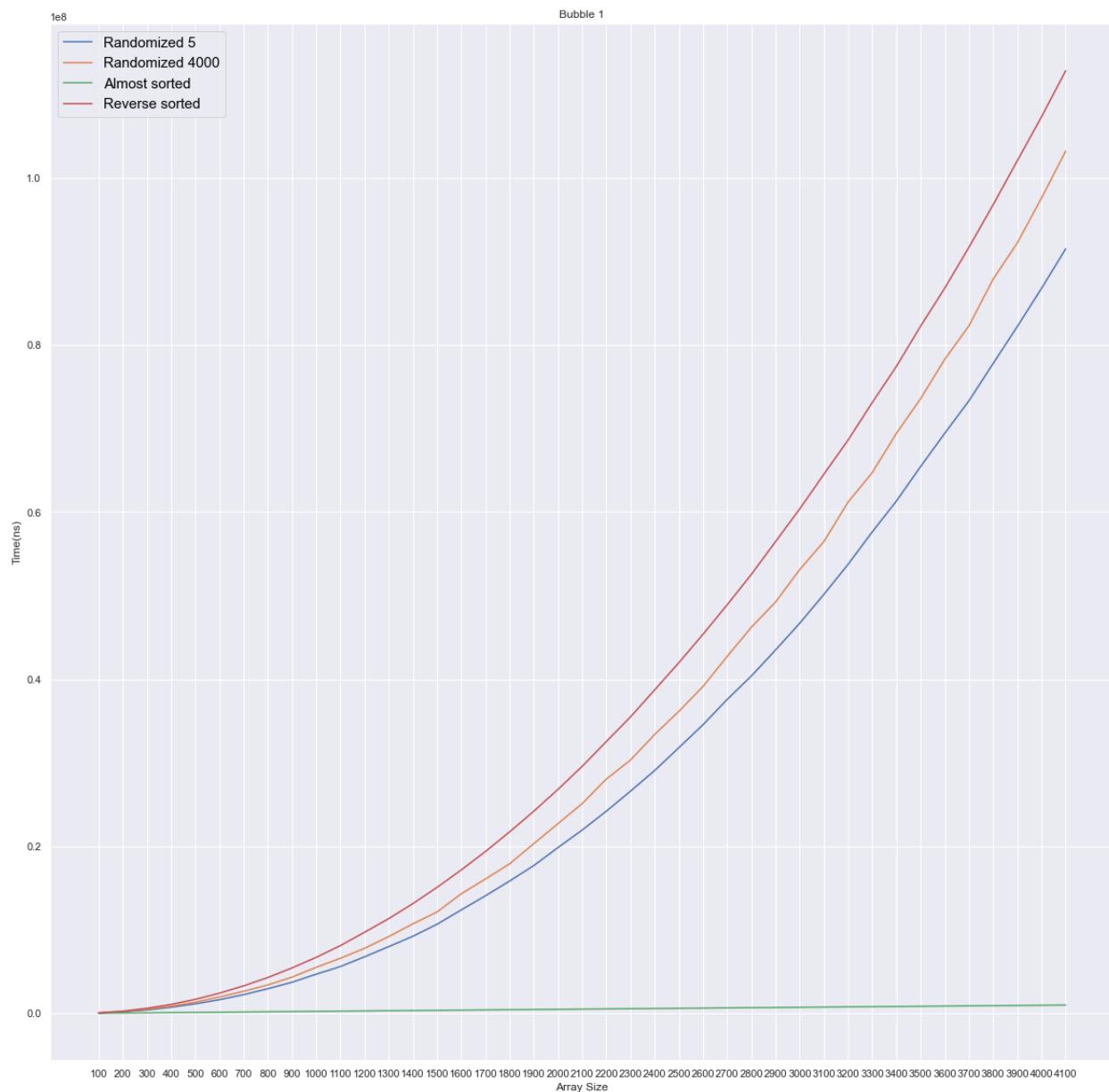
По четырём типам массивов быстрее всего работают подсчётом, Шелла, Шелла-Циура. На случайных значениях все виды пузырька проигрывают, и лишь на почти отсортированном массиве версии с оптимизациями 1 и 1+2 показывают себя приемлемо быстро.

## Пузырьком



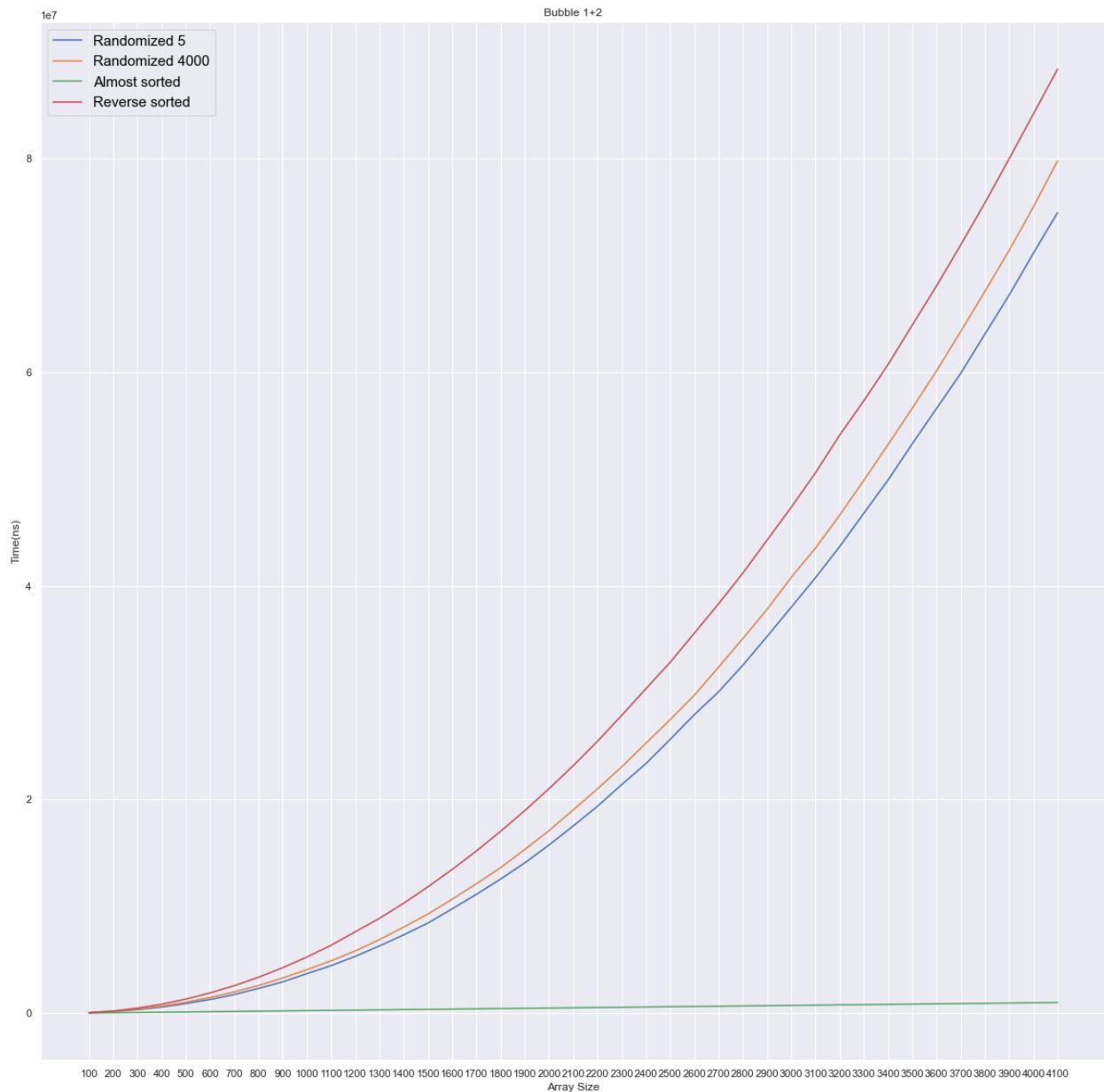
Как и ожидалось, худшим образом себя проявляет на обратно отсортированном массиве, в то время как на почти отсортированном меньше всего приходится делать перестановок.

## Пузырьком с оптимизацией Айверсона 1



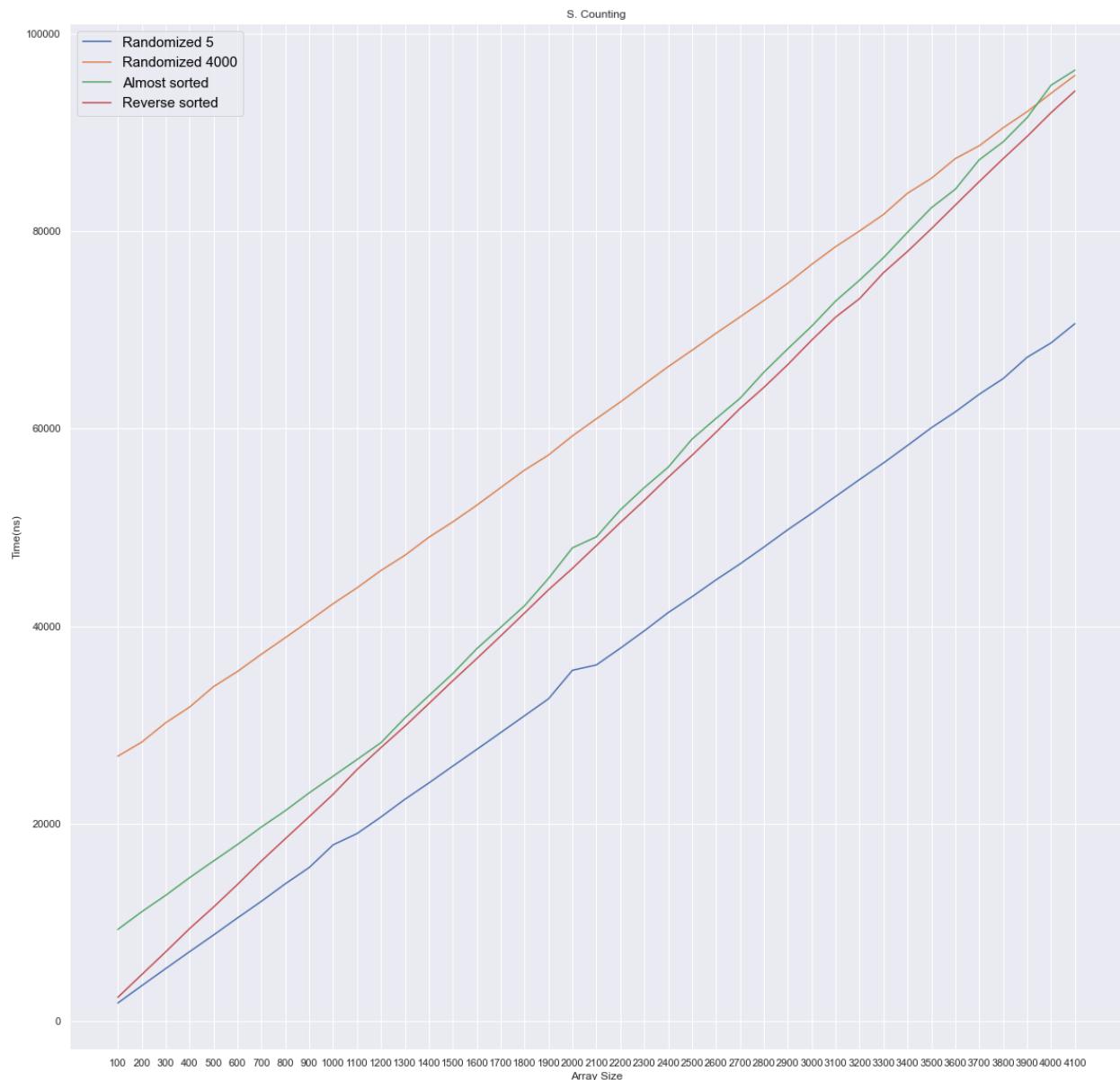
Всё аналогично предыдущему результату, однако у всех графиков время работы намного ниже, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Пузырьком с оптимизацией Айверсона 1+2



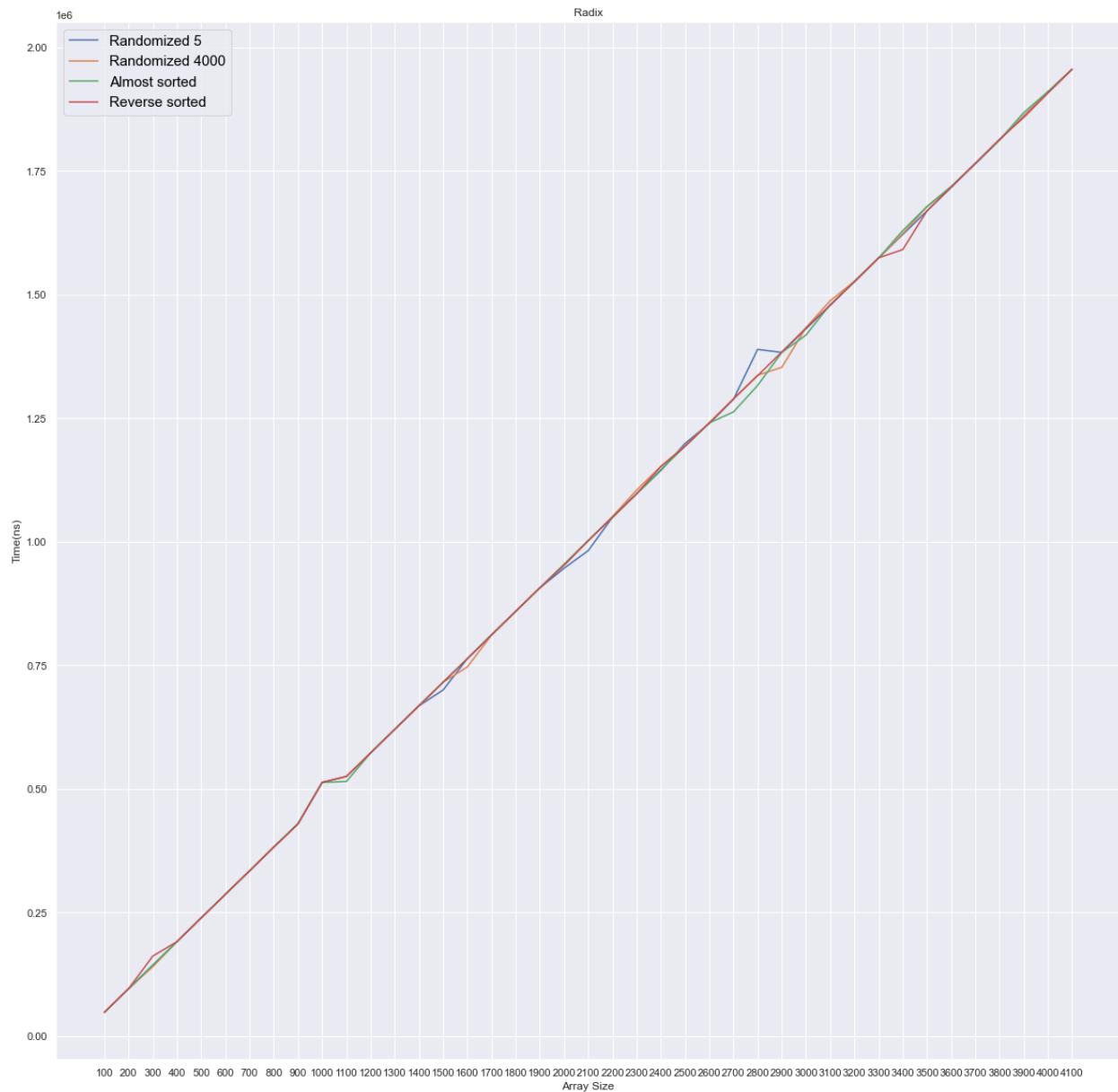
Всё аналогично предыдущему результату, однако у всех графиков время работы намного ниже, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Подсчётом



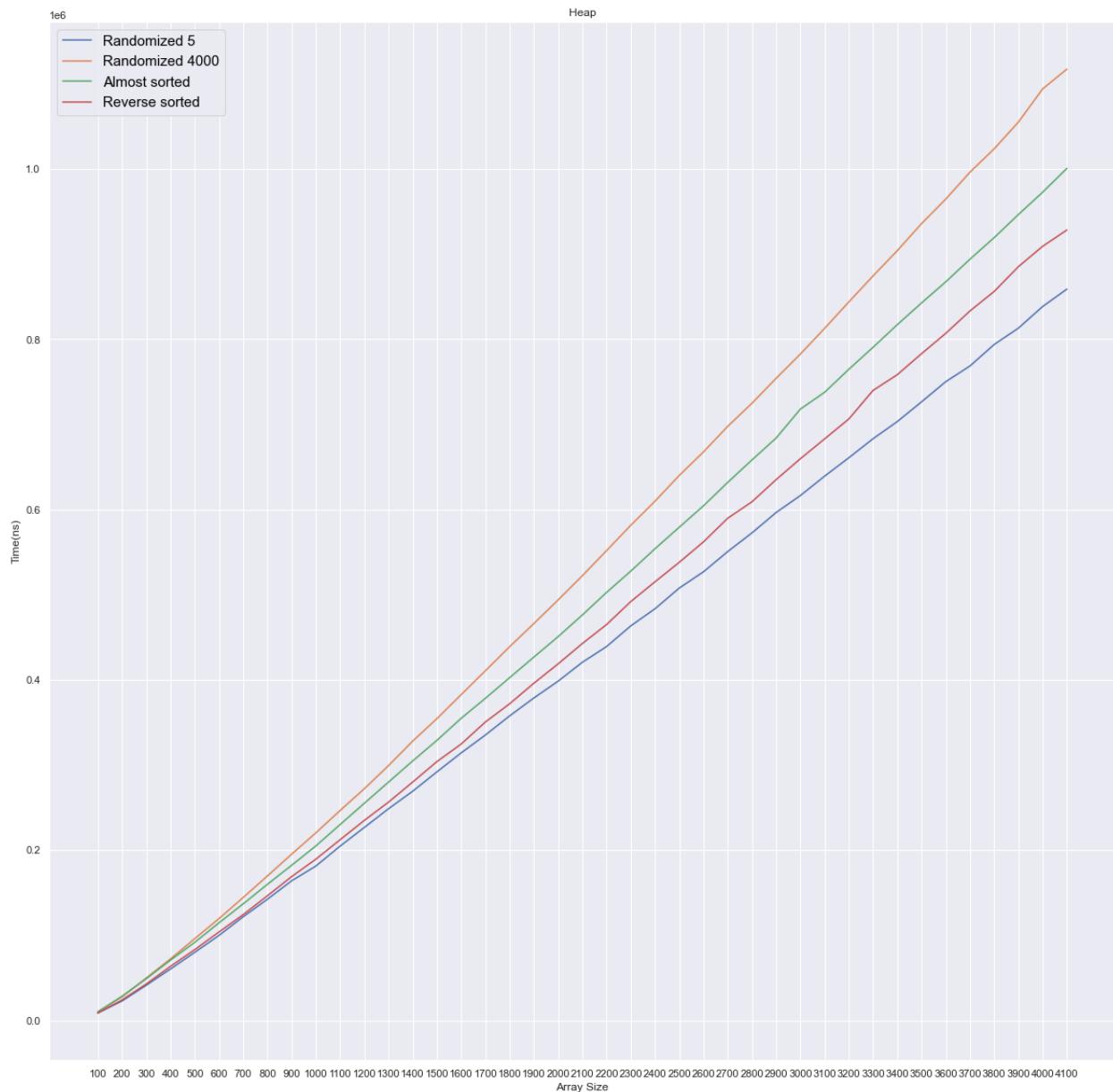
Прямо и обратно сортированные массивы с увеличением размера начинают сортироваться всё медленнее и медленнее, что, скорее всего, обусловлено тем, что при их генерации диапазон равен размеру, и следовательно скорость падает.

## Цифровая



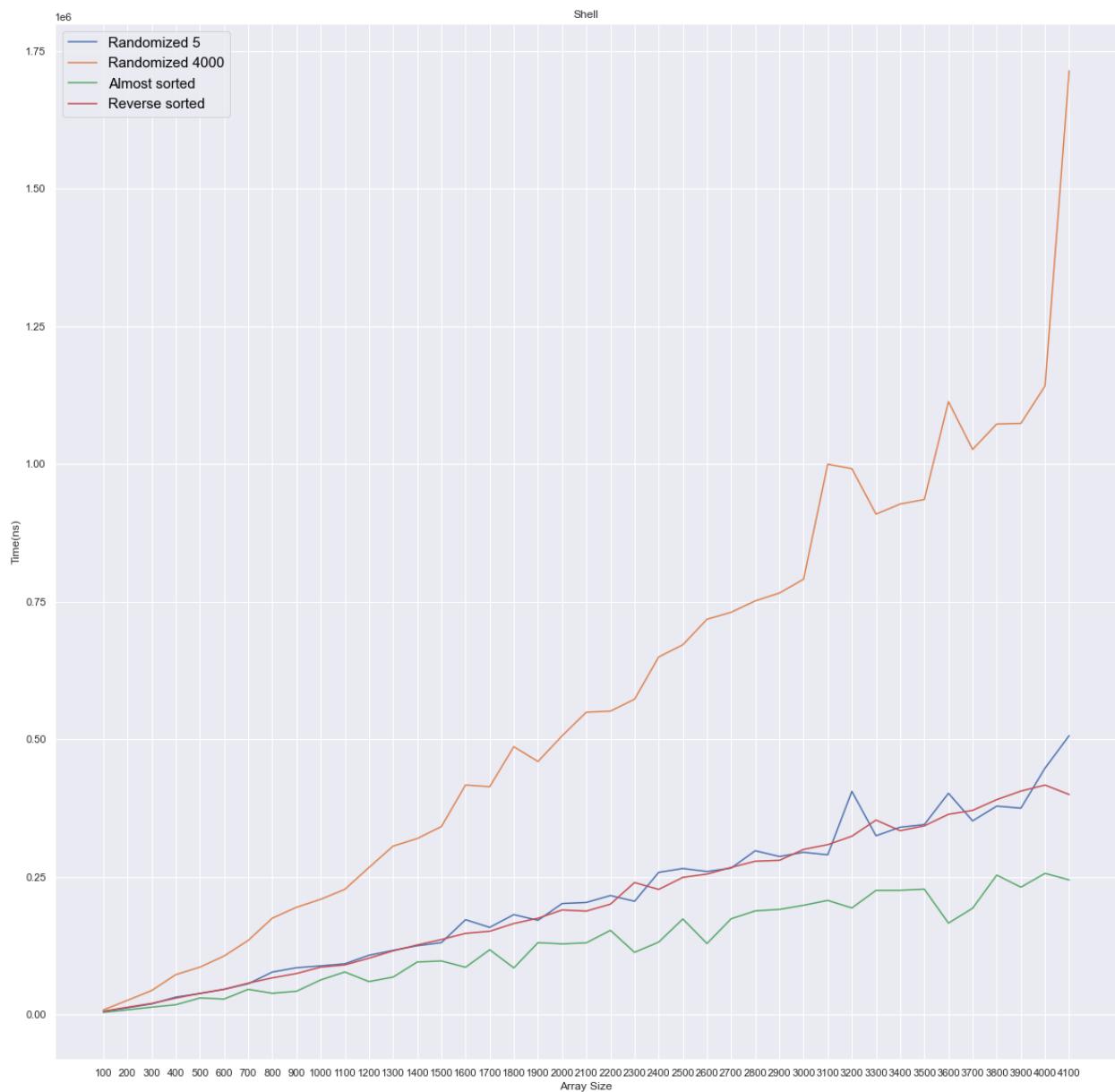
Видим, что цифровая сортировка работает линейно и одинаково для всех типов массивов.

## Пирамидальная



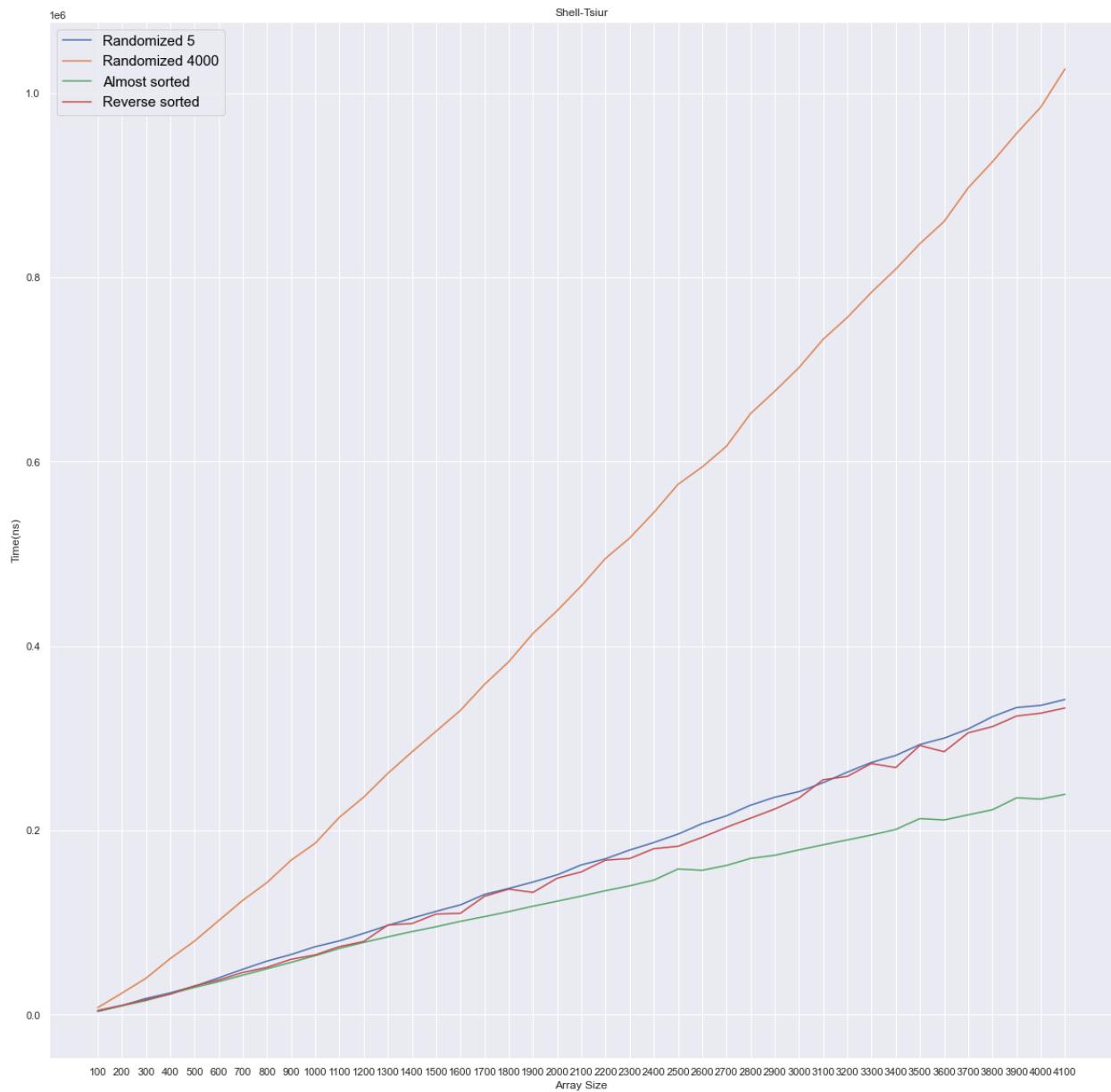
С увеличением размера массива графики больше отдаляются друг от друга. Быстрее всего сортируется рандомизированный массив 5, медленнее всего — рандомизированный массив 4000.

## Шелла



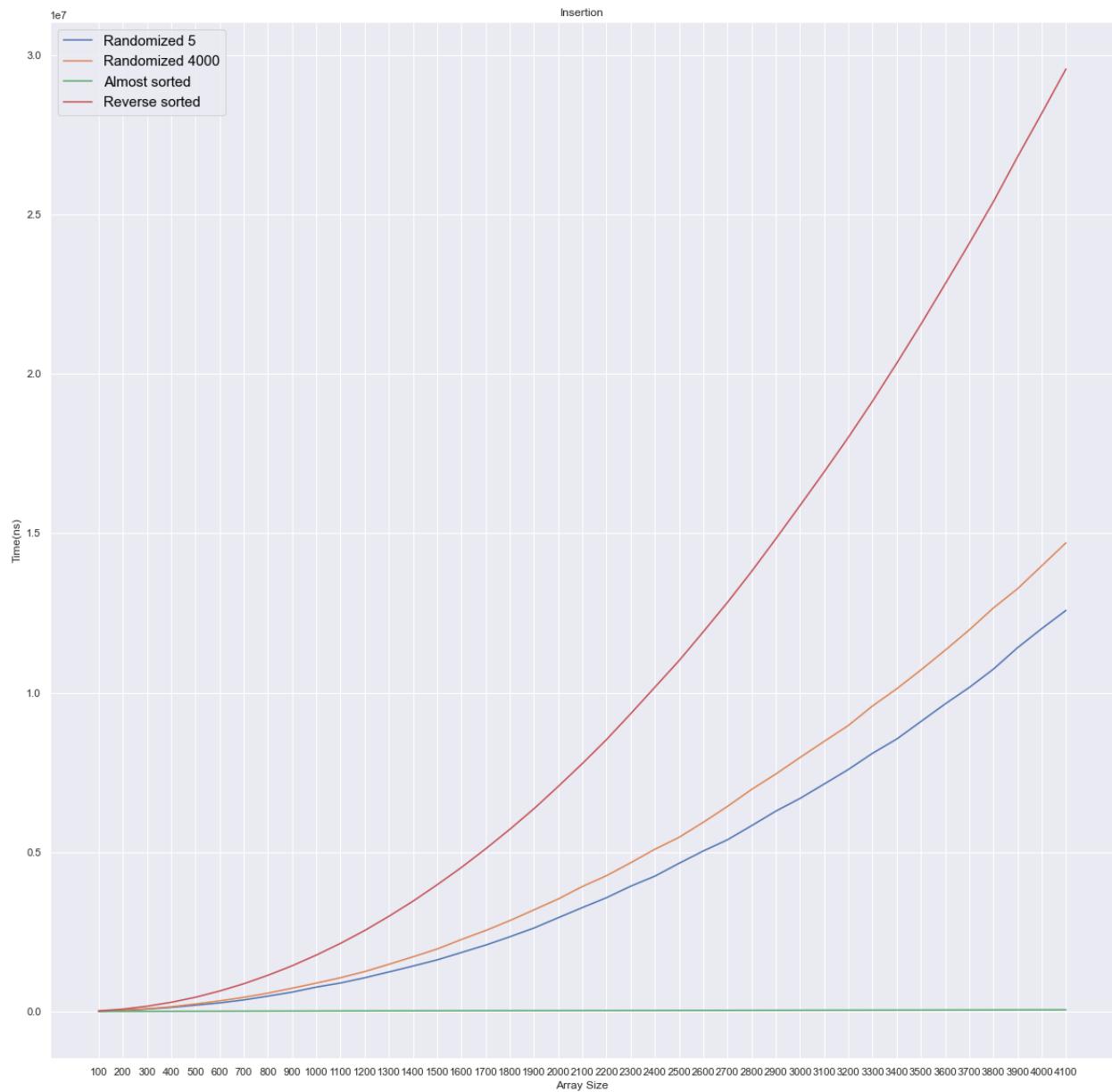
Медленнее всего работает на случайному массиве с диапазоном [1;4000], поскольку слишком часто приходится производить перестановки элементов, быстрее всего — на почти отсортированном.

## Шелла-Циура



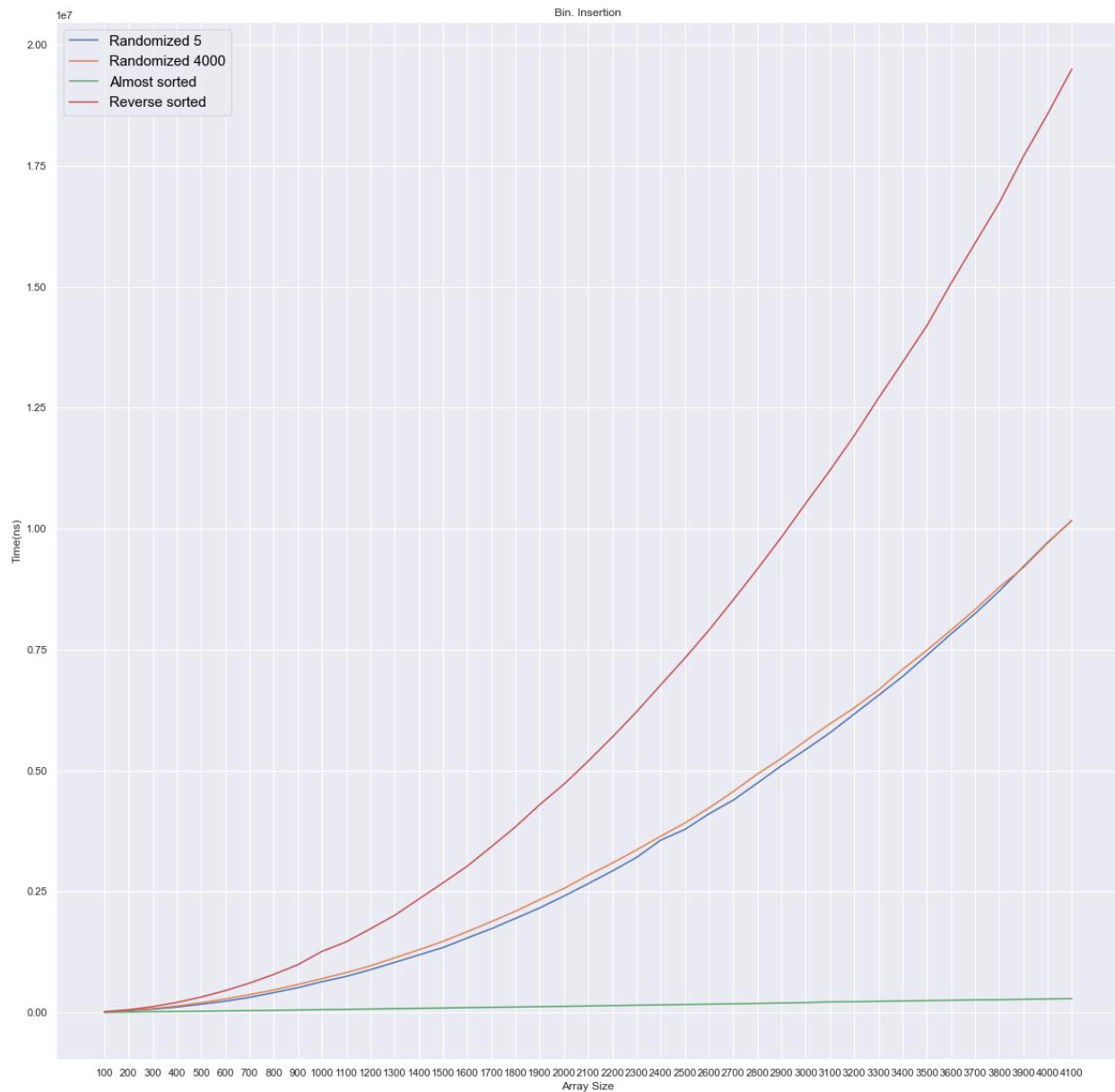
В отличии от предыдущей ситуации, здесь в целом сократилось время работы у каждого типа массива, при этом синий и зелёный графики стали заметно меньше колебаться при увеличении размера массива.

## Вставками



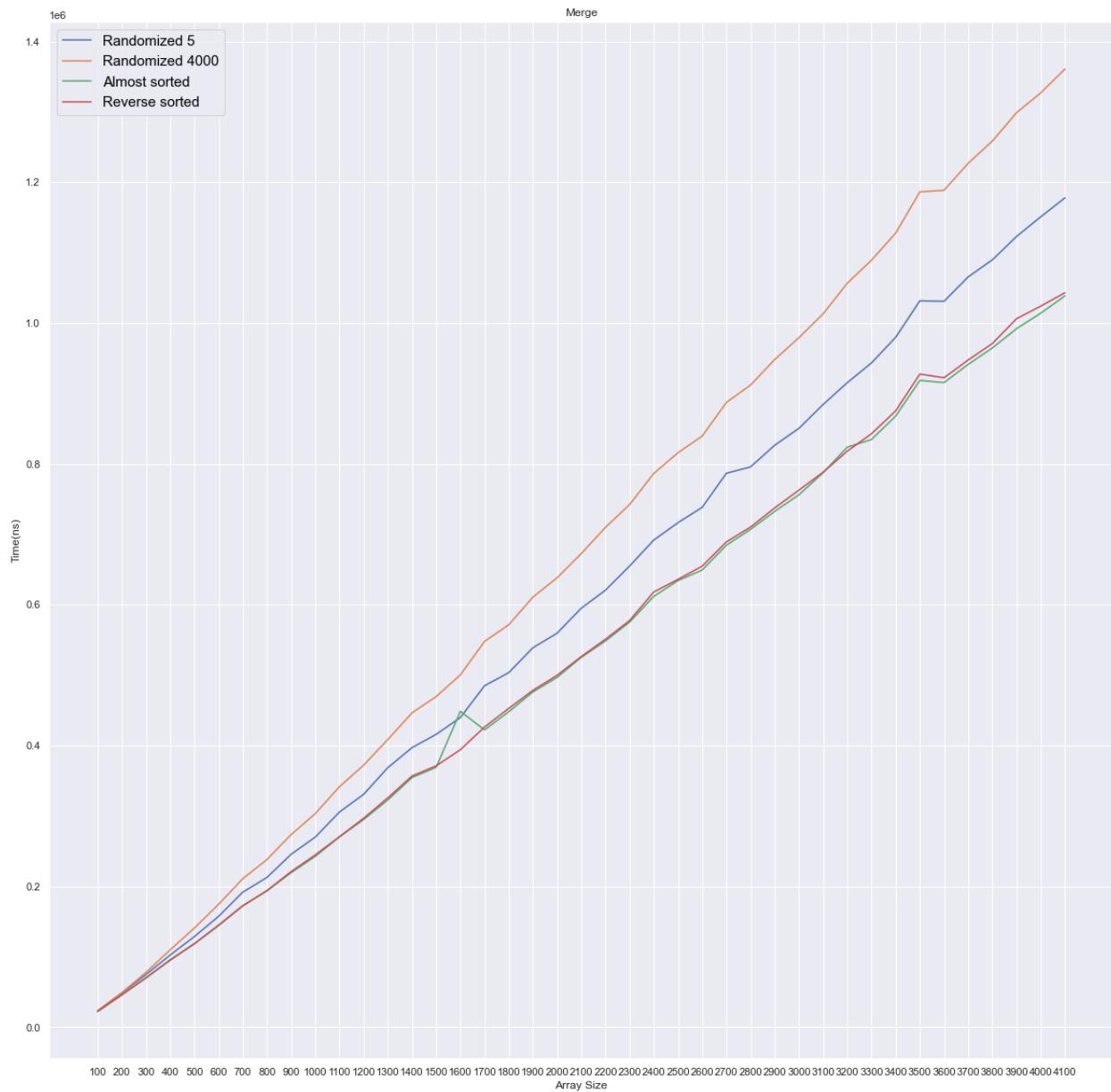
Как и ожидалось, очень медленно работает с обратно отсортированным массивом, поскольку каждый раз приходится проходить весь отрезок до начала, прежде чем вставить элемент на нужную позицию (`array[0]`). Лучше всего показала себя при работе с почти отсортированным, поскольку в большинстве случаев элемент не приходится вставлять в другую позицию уже отсортированного подотрезка.

## Бинарными вставками



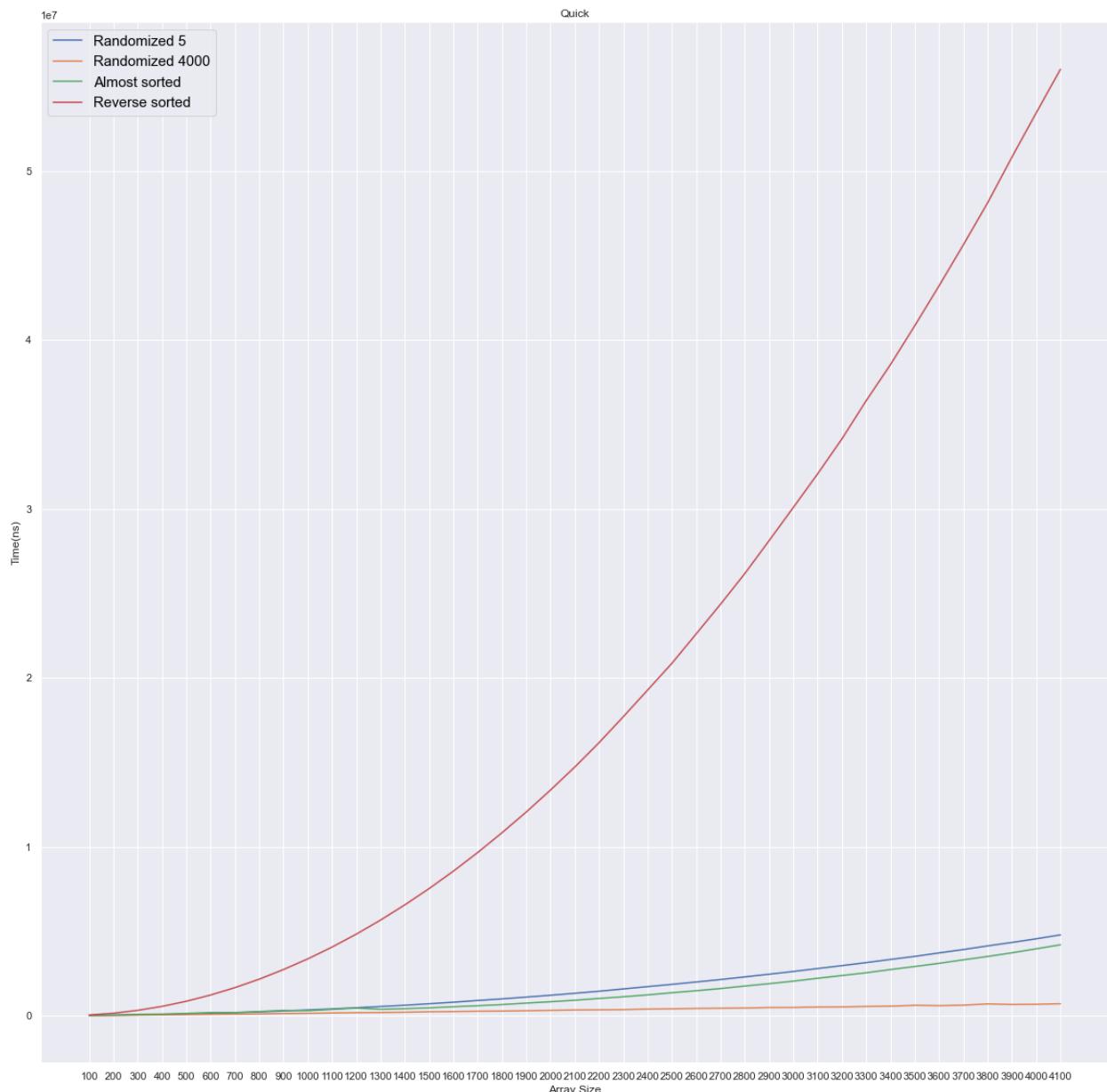
Время работы всех массивов, кроме почти отсортированного, немного сократилось, а так наблюдается ситуация, аналогичная предыдущему случаю.

## **Слиянием**



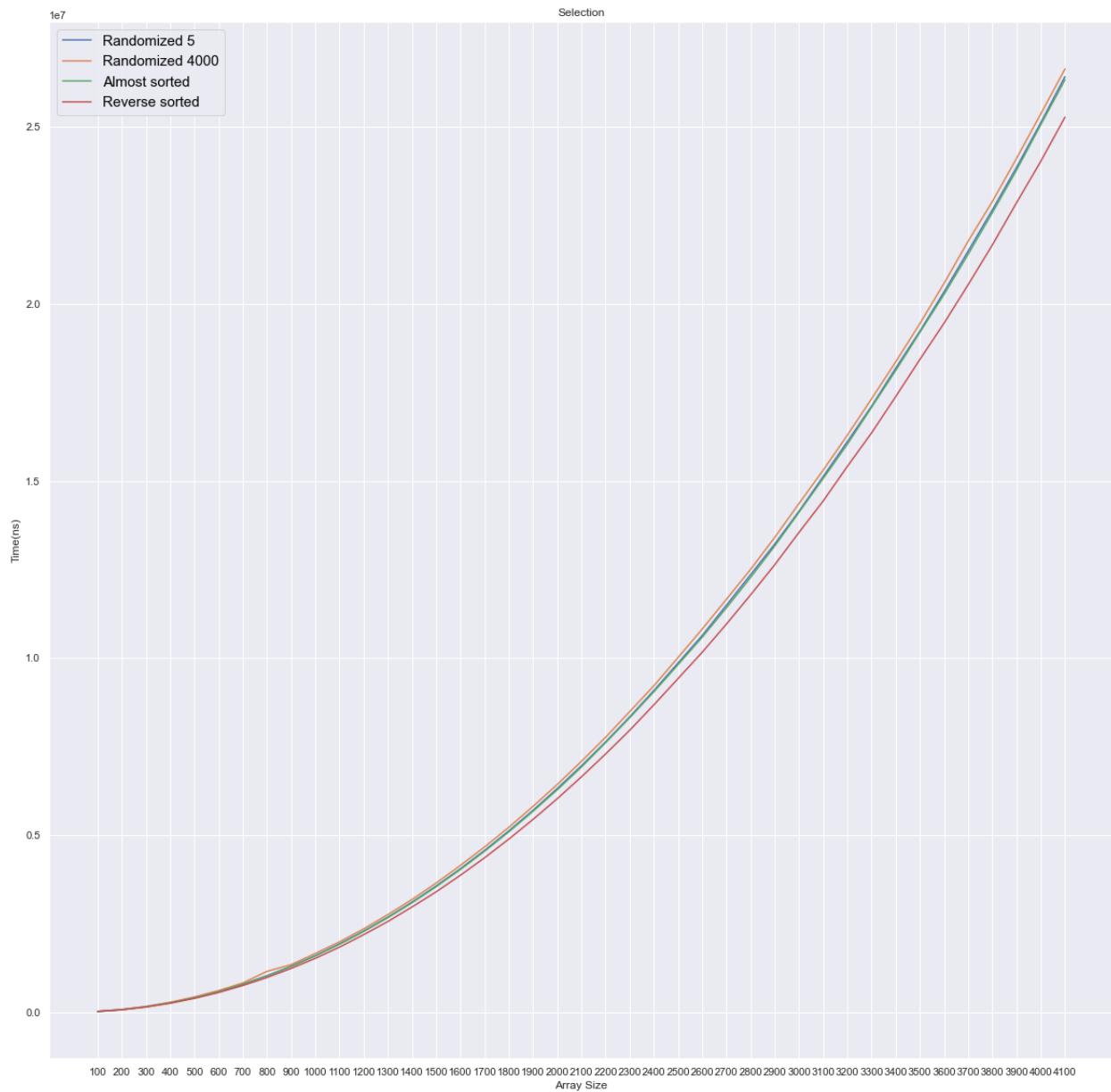
Лучше всего работает на максимально упорядоченных массивах (по возрастанию и убыванию). Графики расходятся с увеличением размера массивов.

## Быстрая



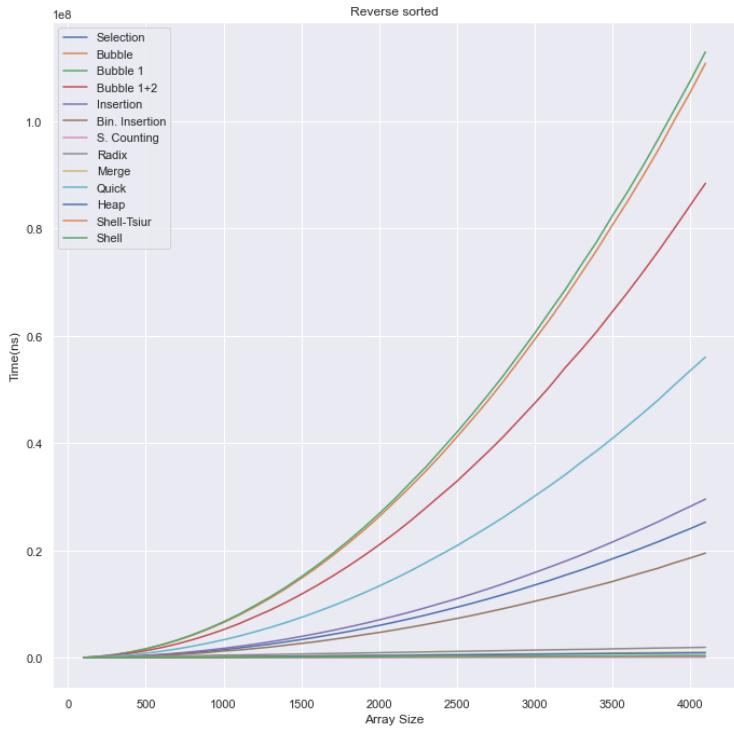
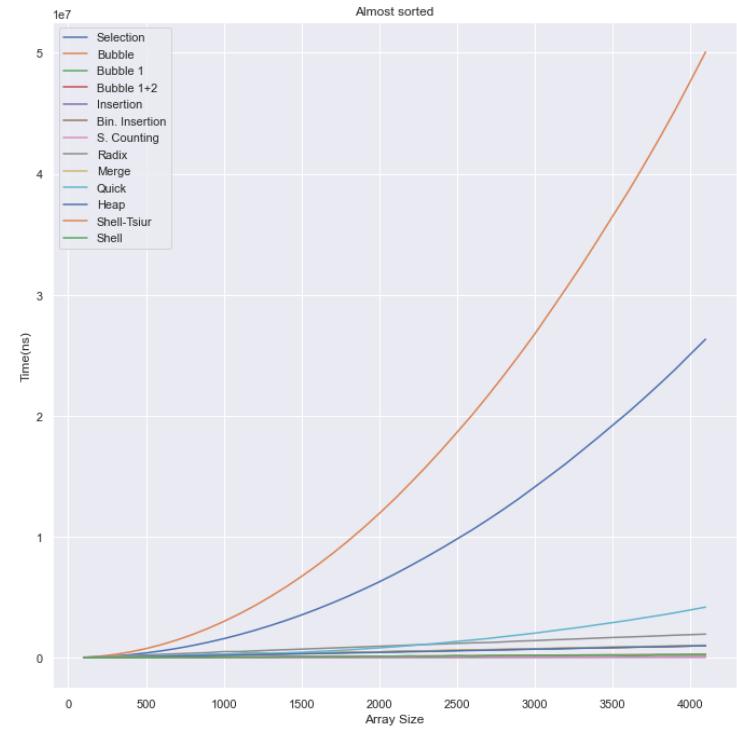
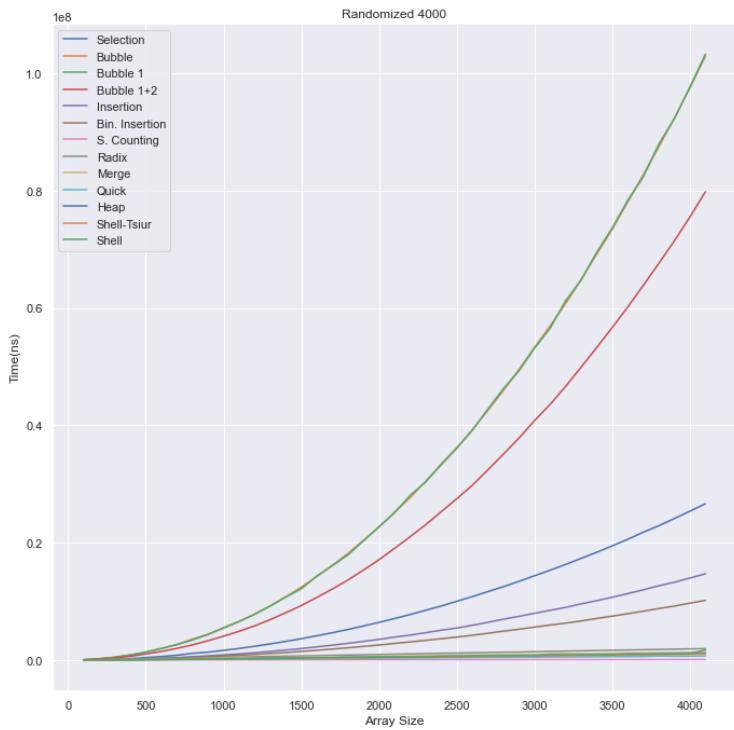
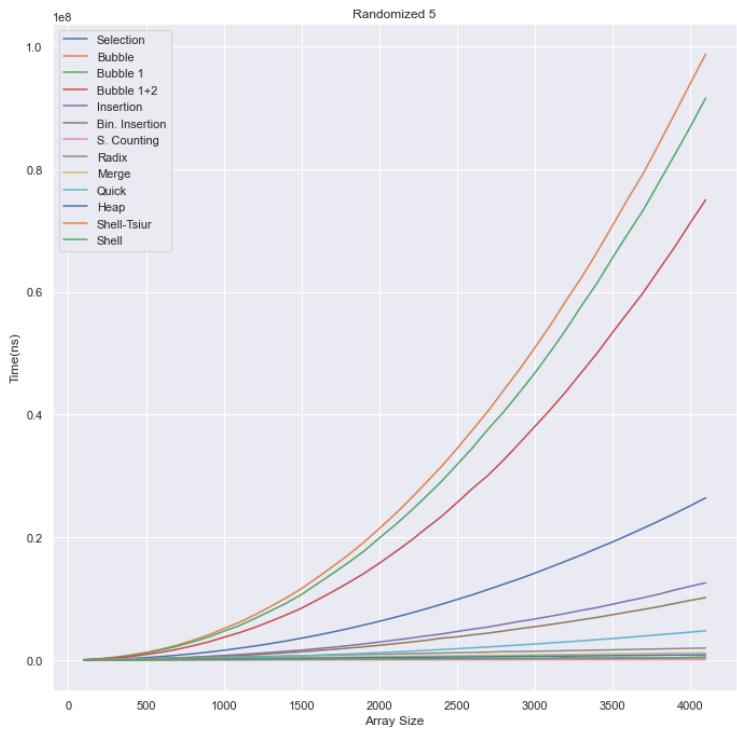
Очень медленно работает на обратно отсортированном массиве, поскольку за pivot берётся первый элемент подотрезка (т. е. максимальный для текущего подотрезка).

## Выбором



Почти одинаково работает на всём диапазоне размера массива.

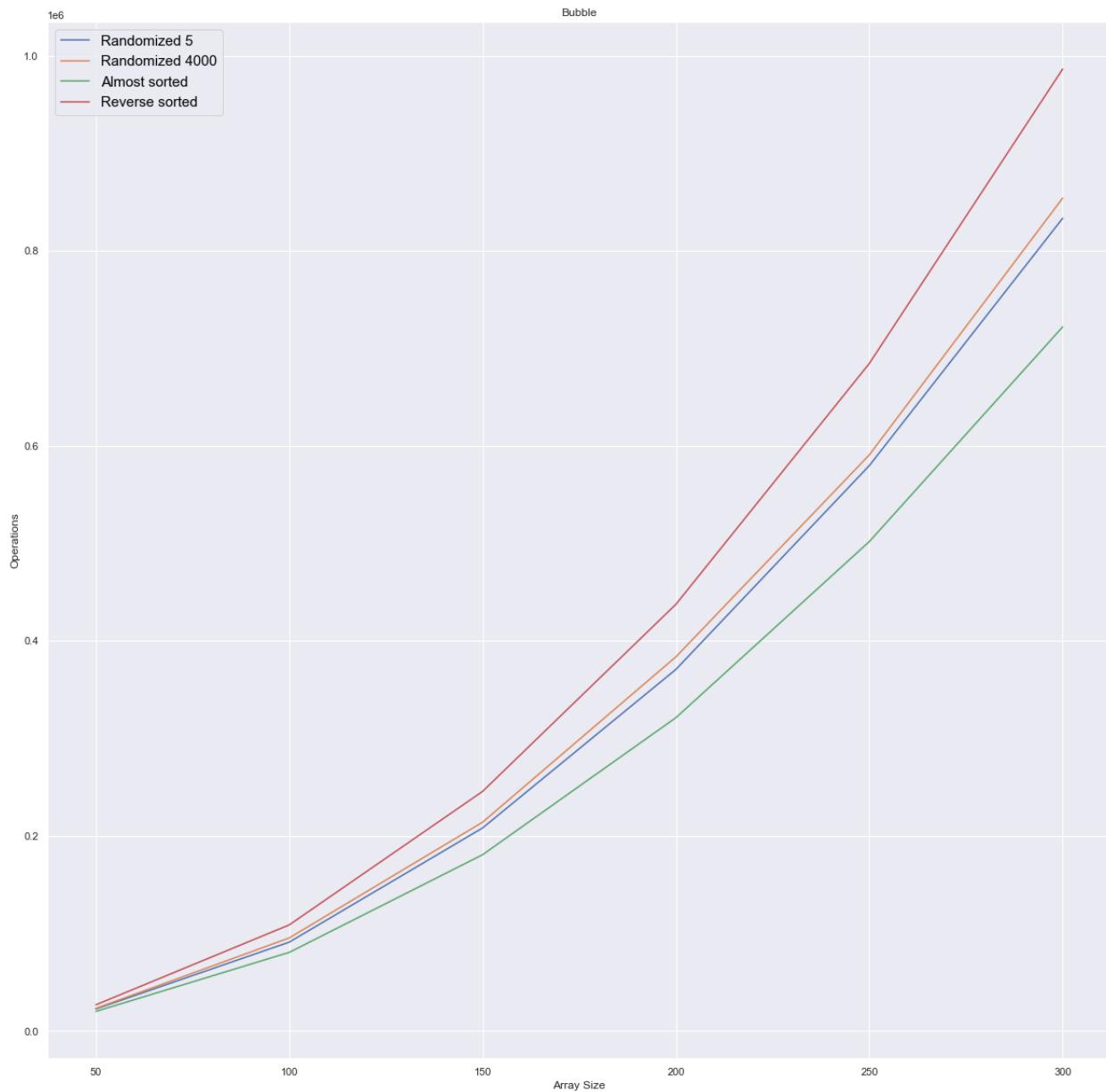
# Сравнение сортировок



По четырём типам массивов быстрее всего работают подсчётом, цифровая, Шелла, Шелла-Циура. На случайных значениях все виды пузырька проигрывают, и лишь на почти отсортированном массиве версии с оптимизациями 1 и 1+2 показывают себя приемлемо быстро.

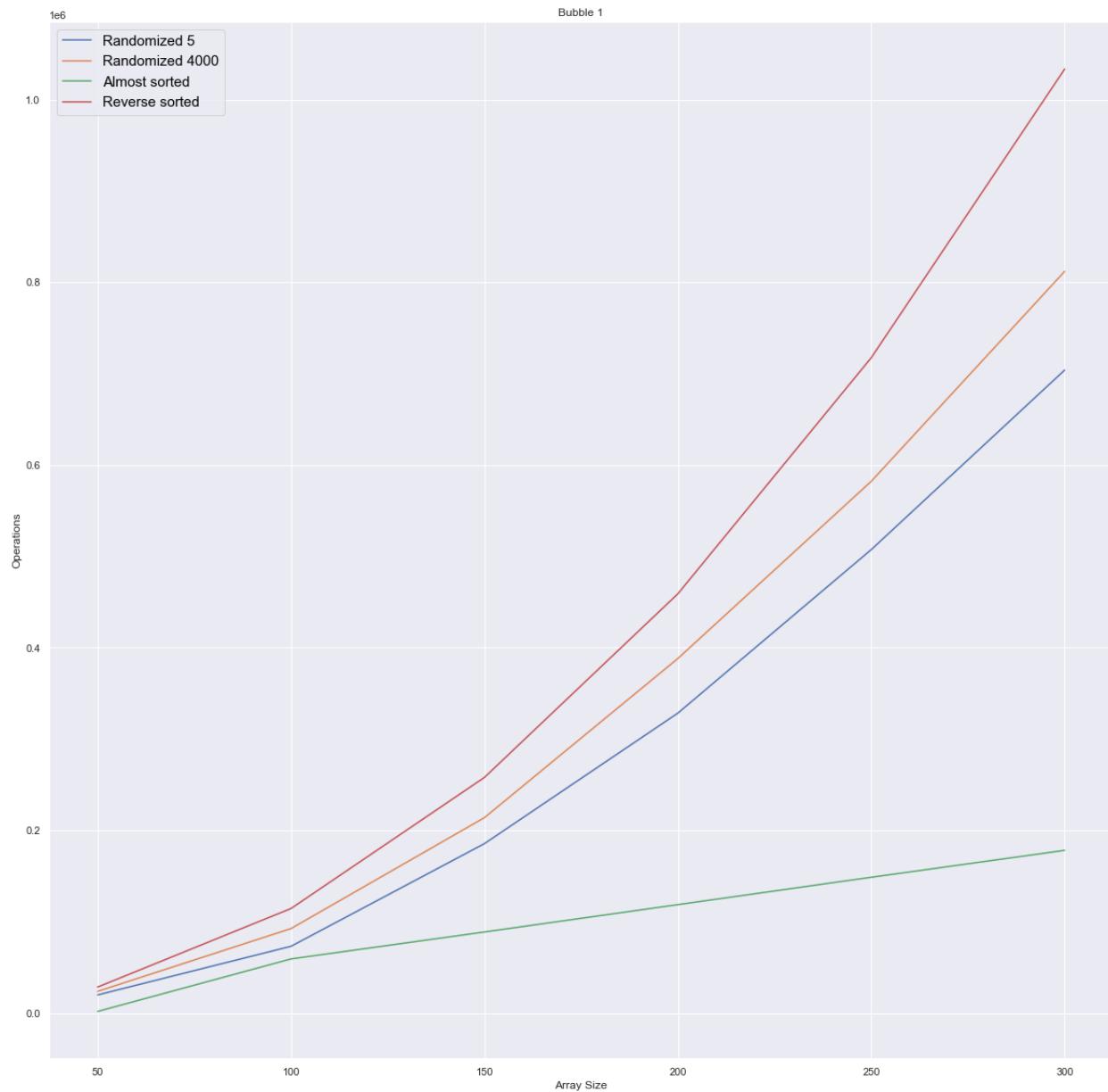
# Подсчёт количества операций

## Пузырьком



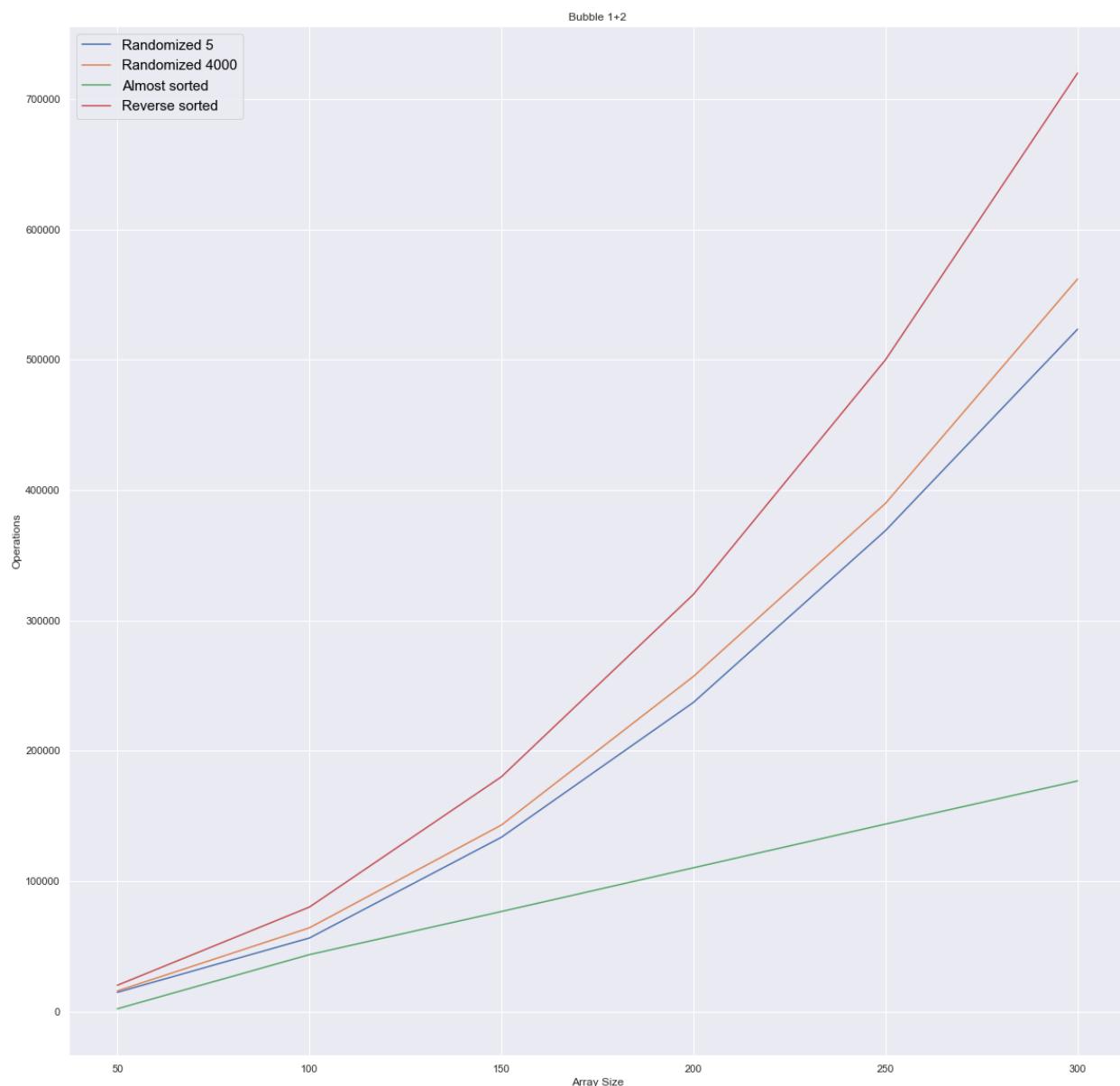
Как и ожидалось, худшим образом себя проявляет на обратно отсортированном массиве, в то время как на почти отсортированном меньше всего приходится делать перестановок.

## Пузырьком с оптимизацией Айверсона 1



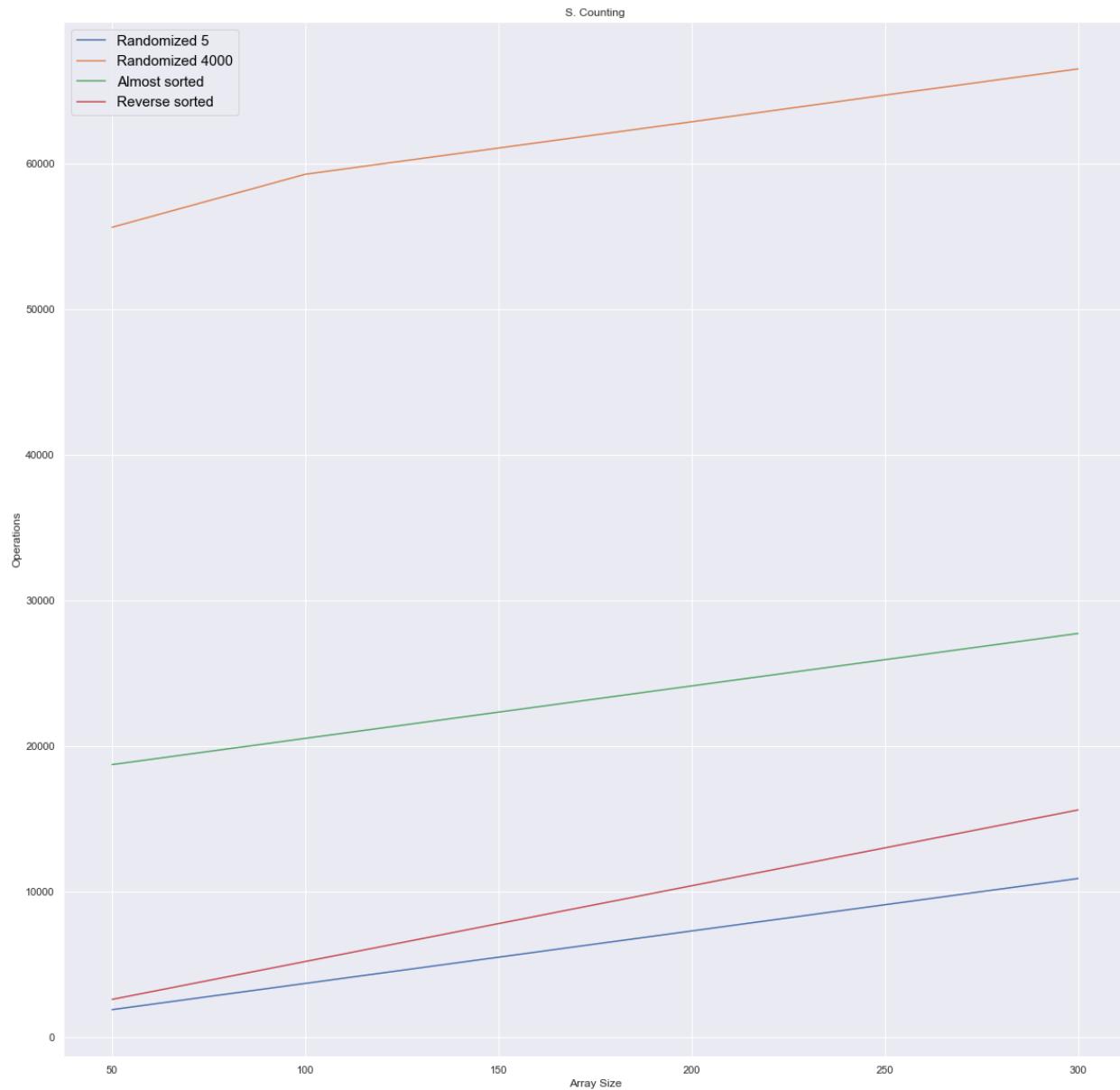
Всё аналогично предыдущему результату, однако у всех графиков кол-во операций намного меньше, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## **Пузырьком с оптимизацией Айверсона 1+2**



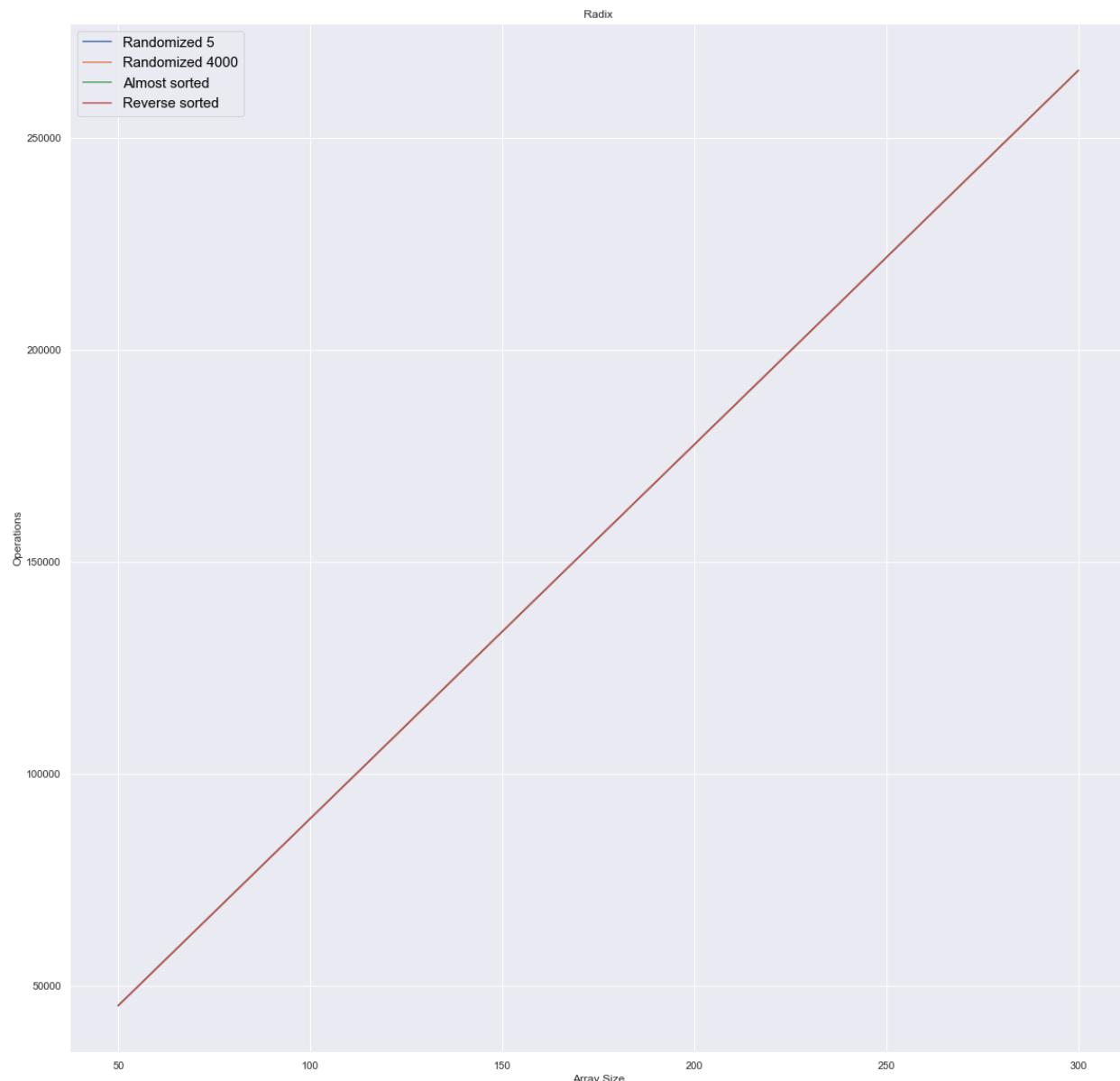
Всё аналогично предыдущему результату, однако у всех графиков кол-во операций намного меньше, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Подсчётом



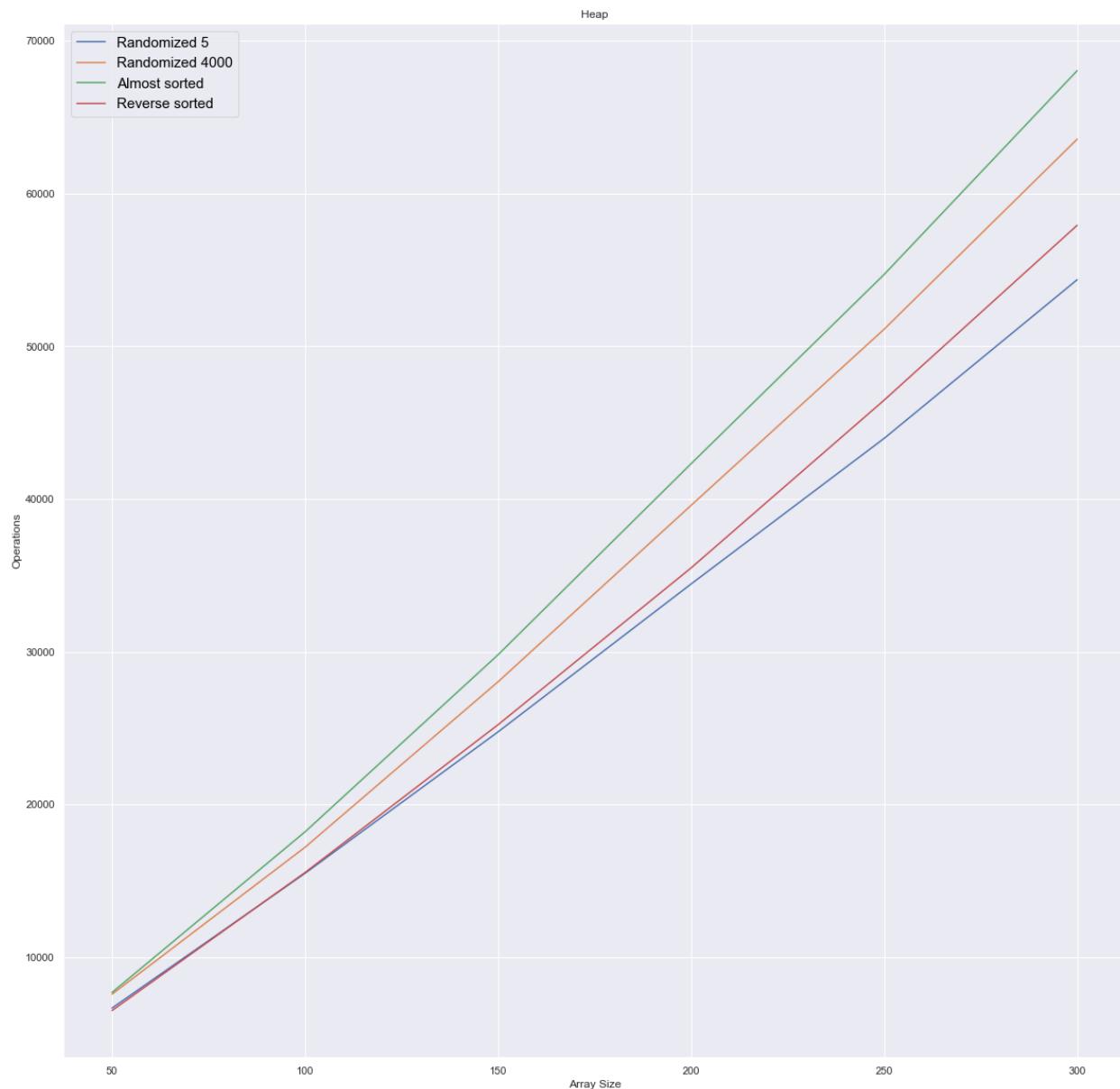
Как и ожидалось, хуже всего работает на числах с большим разбросом (модуль разности максимума и минимума), поэтому синий график самый низкий по операциям, а жёлтый — самый высокий.

## Цифровая



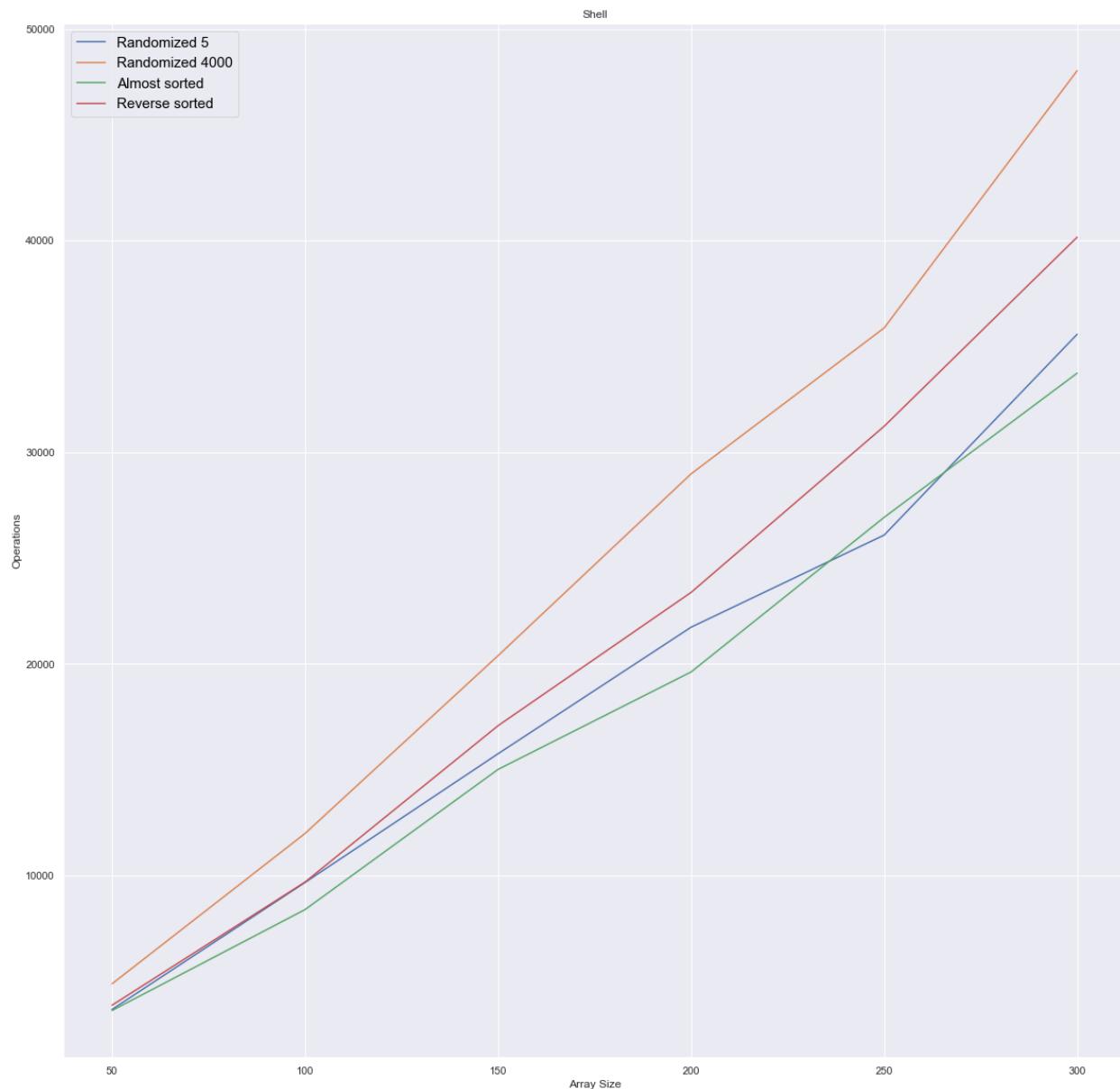
Видим, что цифровая сортировка работает линейно и одинаково для всех типов массивов.

## Пирамидальная



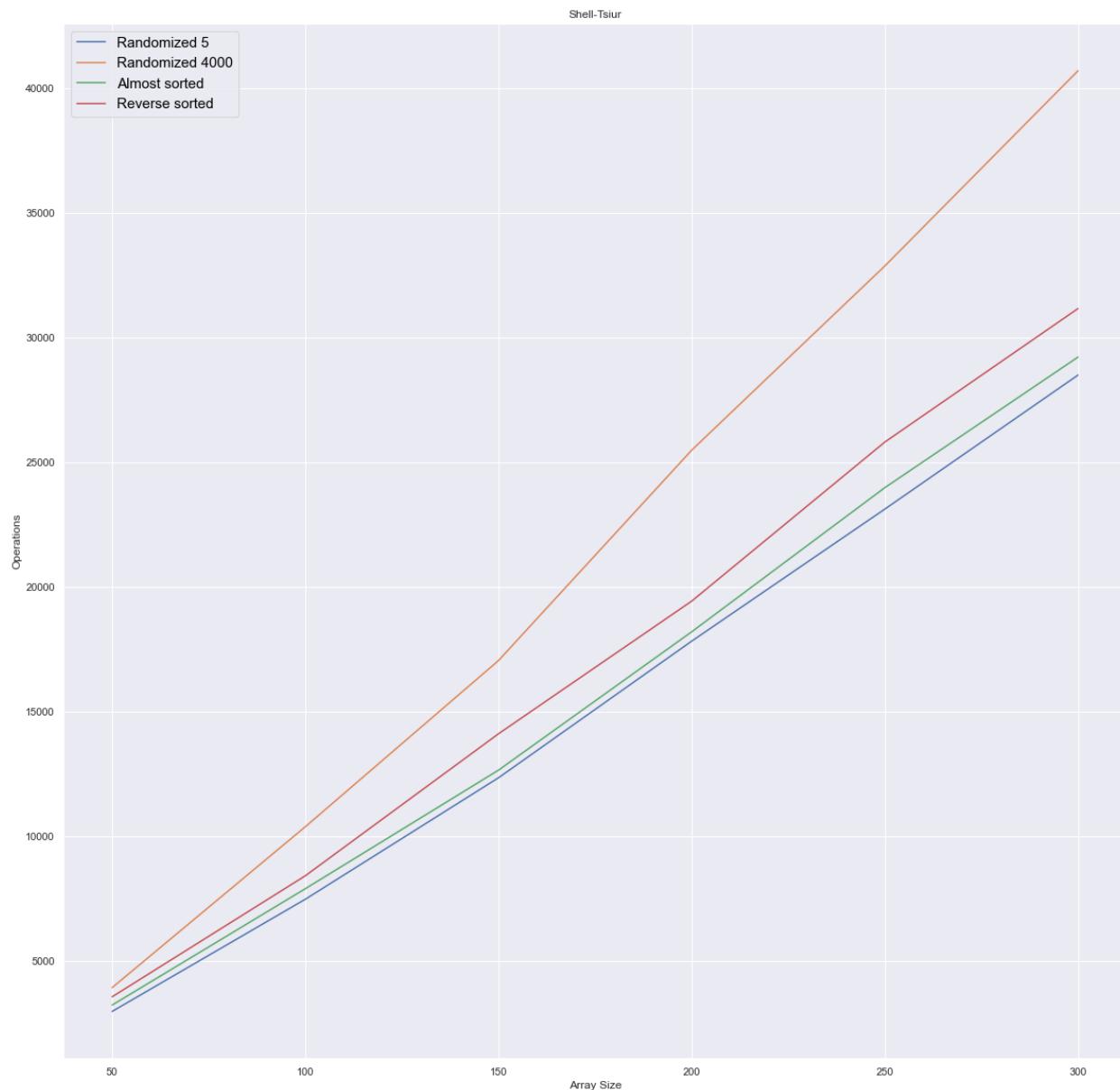
Лучше работает на обратно отсортированном массиве и с маленьким диапазоном значений.

## Шелла



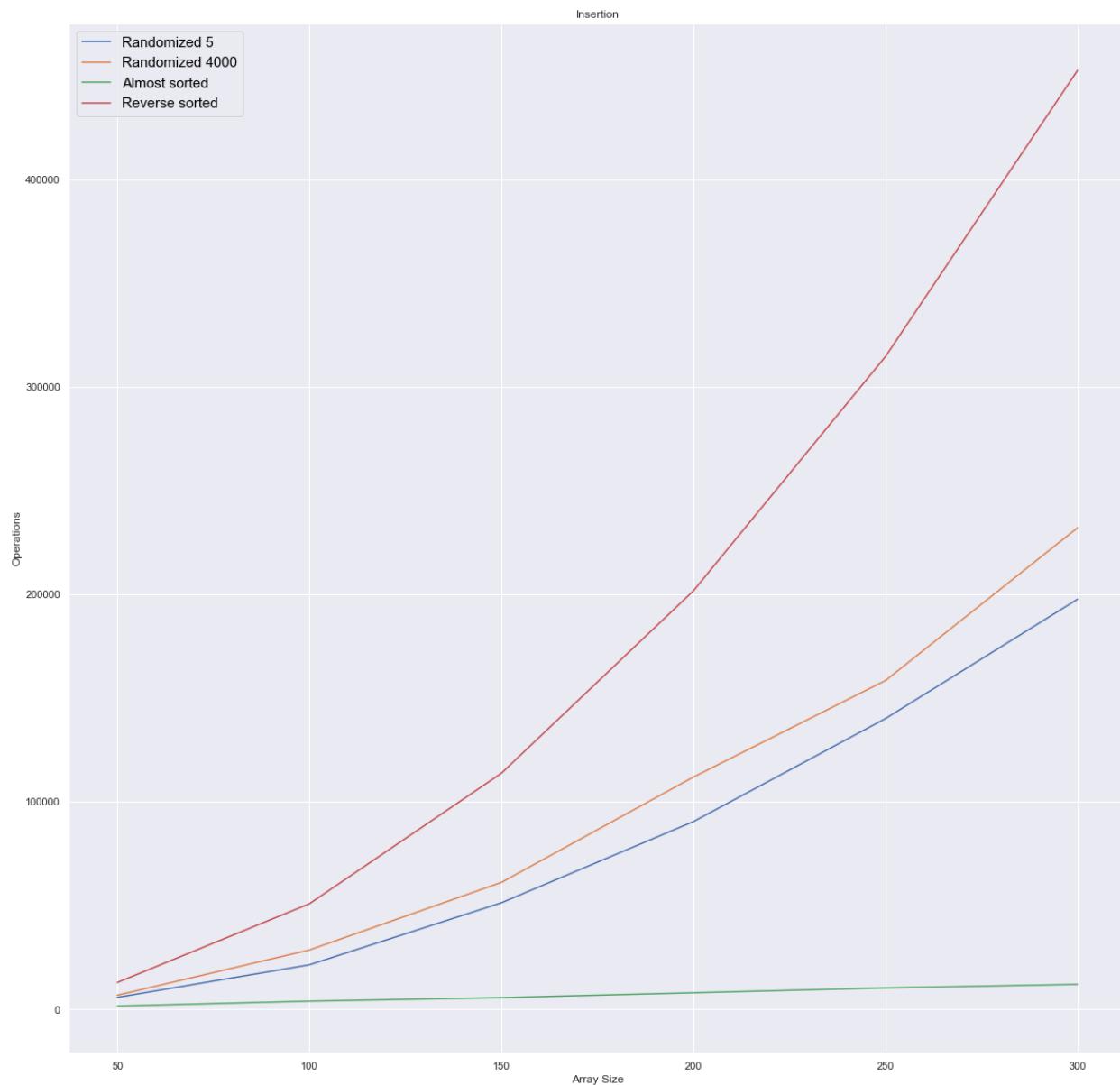
Хуже всего работает на случайном массиве с диапазоном [1;4000], поскольку слишком часто приходится производить перестановки элементов.

## Шелла-Циура



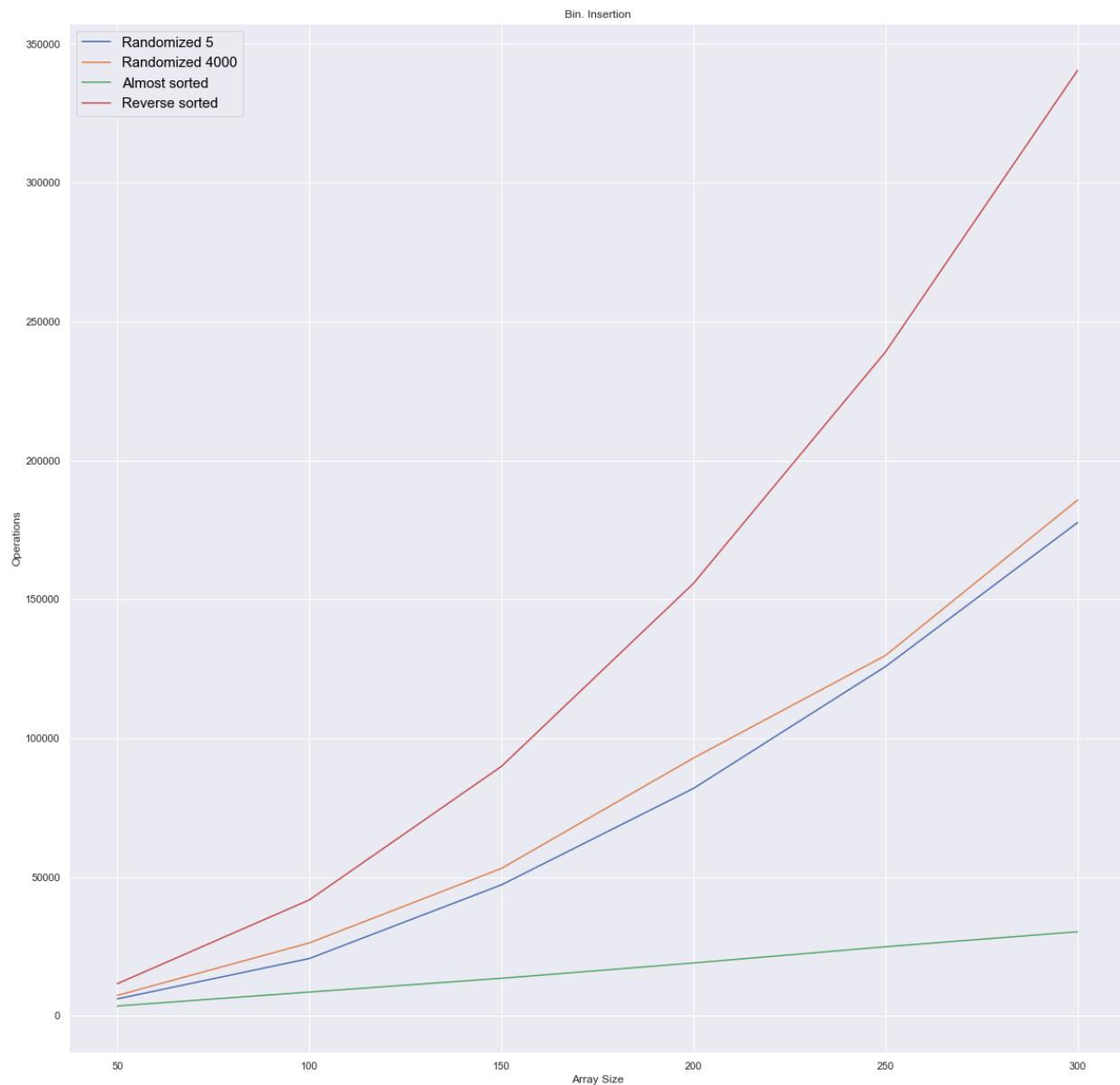
В отличии от предыдущей ситуации, здесь в целом сократилось количество операций у каждого типа массива, при этом синий и зелёный графики стали равномерно идти при увеличении размера массива.

## Вставками



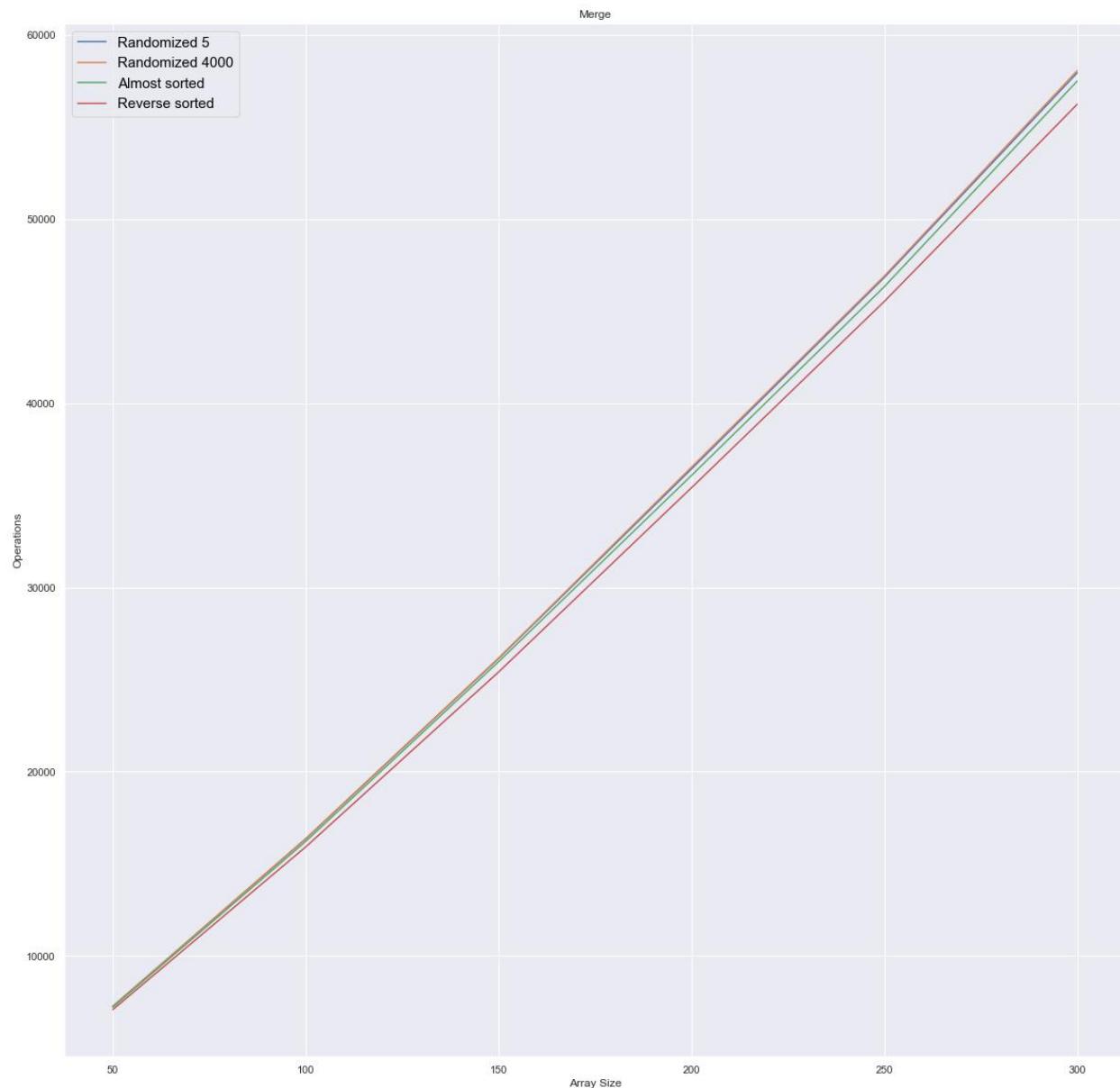
Как и ожидалось, очень много операций производится с обратно отсортированным массивом, поскольку каждый раз приходится проходить весь отрезок до начала, прежде чем вставить элемент на нужную позицию (`array[0]`). Лучше всего показала себя при работе с почти отсортированным, поскольку в большинстве случаев элемент не приходится вставлять в другую позицию уже отсортированного подотрезка.

## **Бинарными вставками**



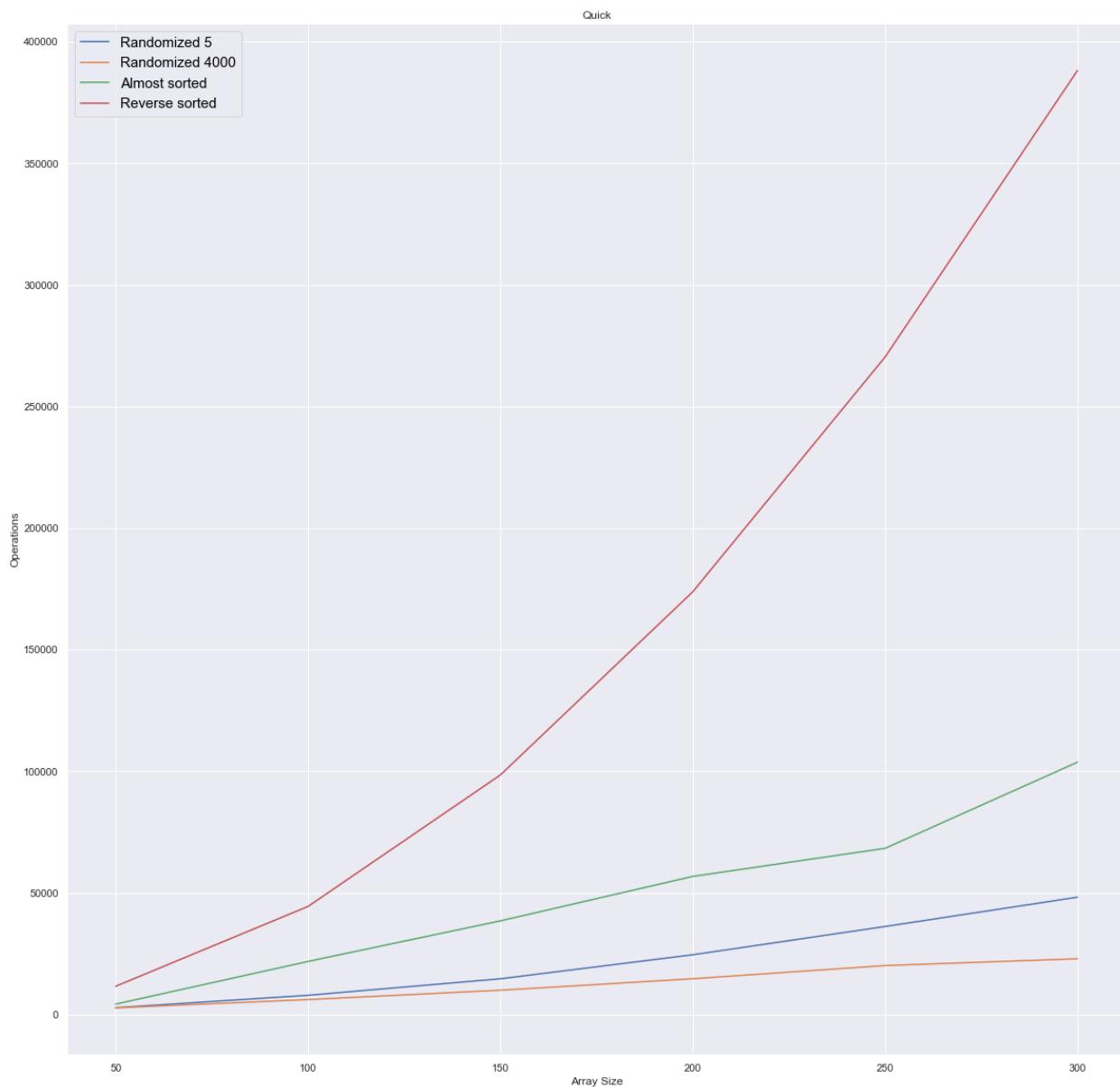
Количество операций у всех массивов немного сократилось, а так наблюдается ситуация, аналогичная предыдущему случаю.

## **Слиянием**



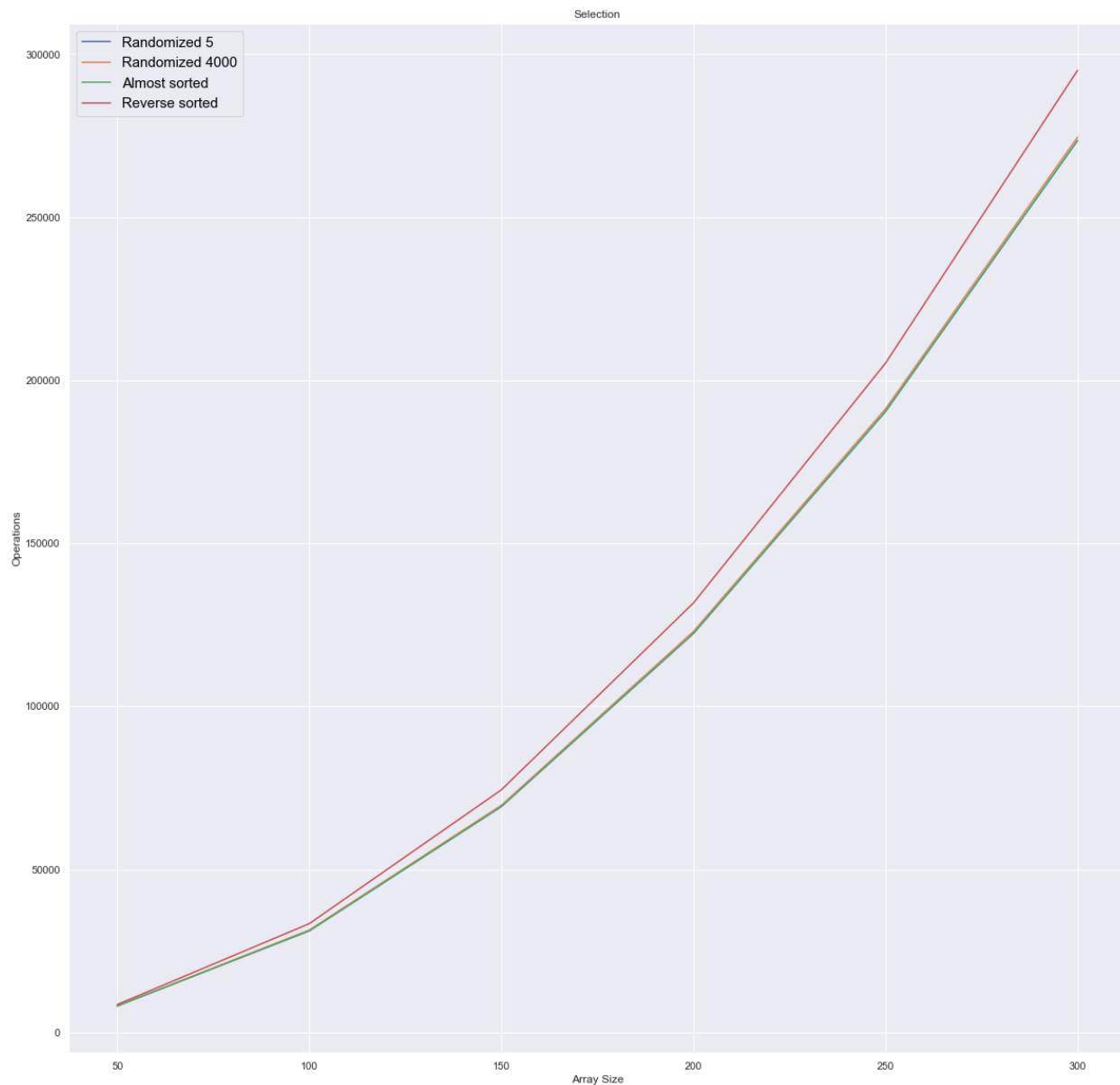
Результаты в целом похожие, немного меньше операций на максимально упорядоченных массивах (по возрастанию и убыванию). Графики незначительно расходятся с увеличением размера массивов.

## **Быстрая**



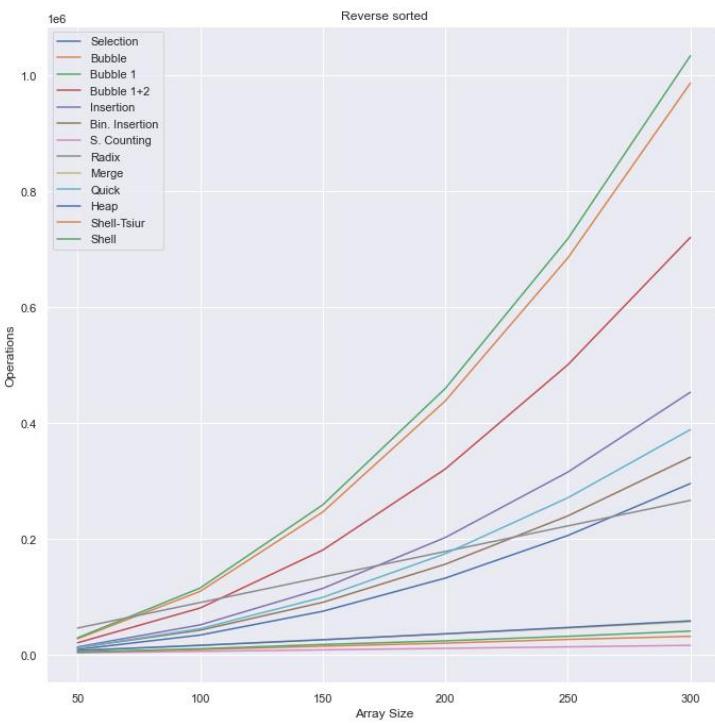
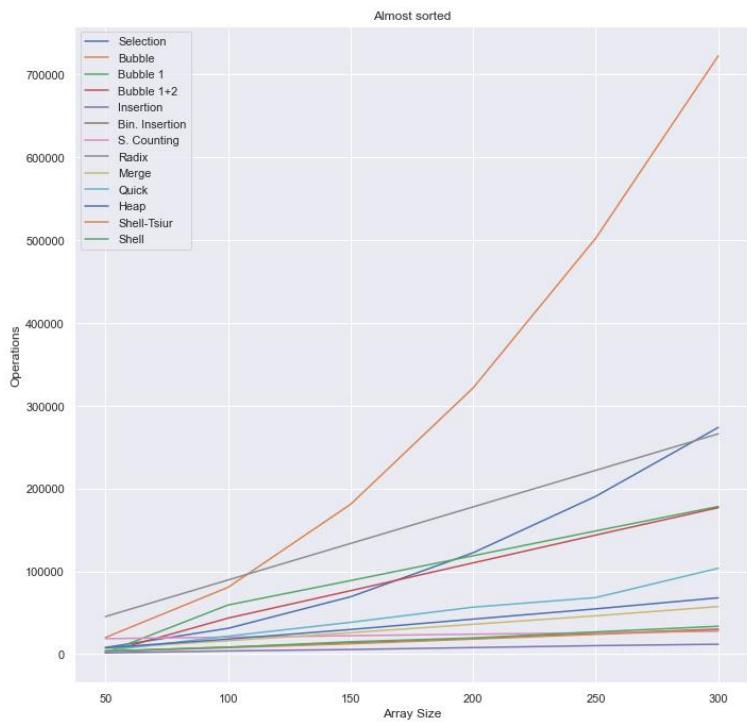
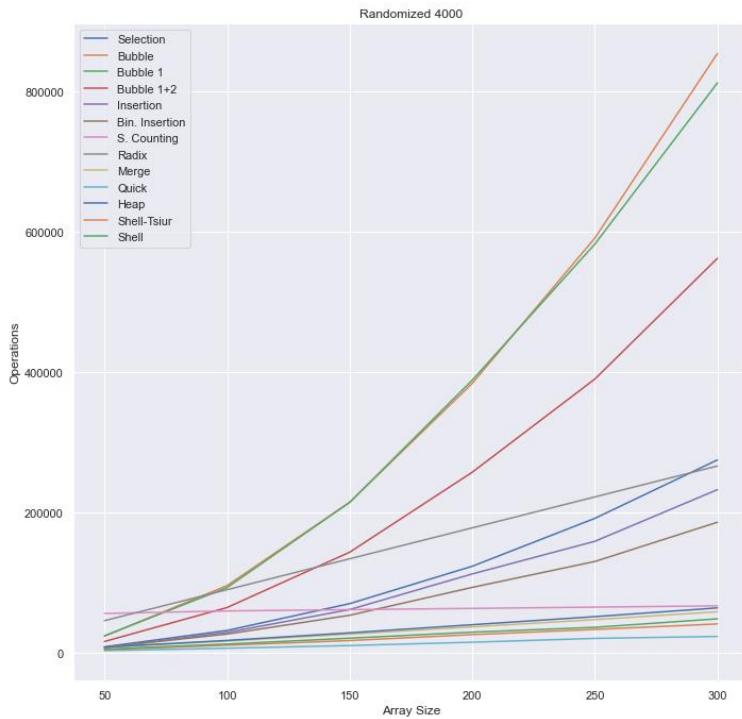
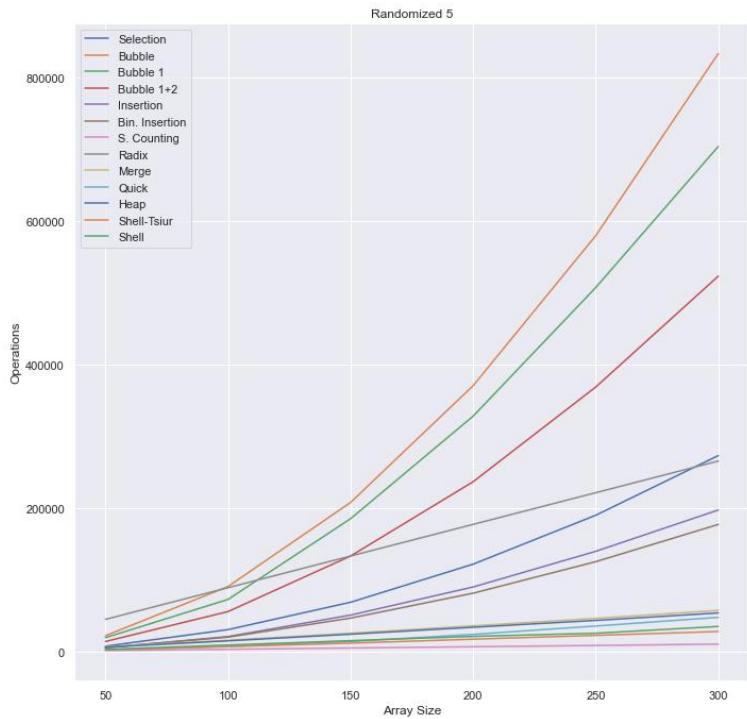
Очень много операций на обратно отсортированном массиве, поскольку за pivot берётся первый элемент подотрезка (т. е. максимальный для текущего подотрезка).

## **Выбором**



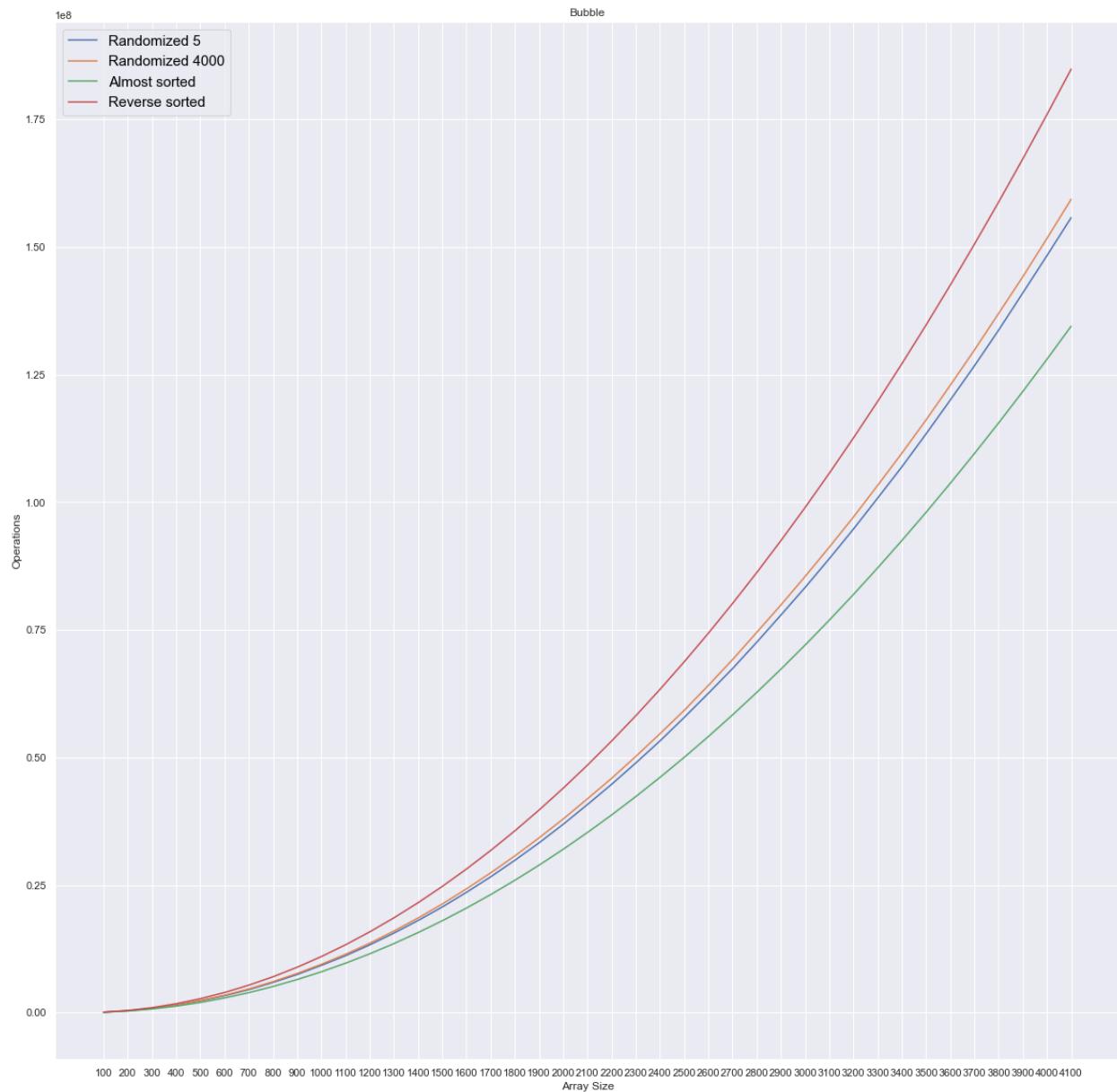
Почти одинаково работает на всём диапазоне размера массива, кроме обратно отсортированного — операций производится больше.

# Сравнение сортировок



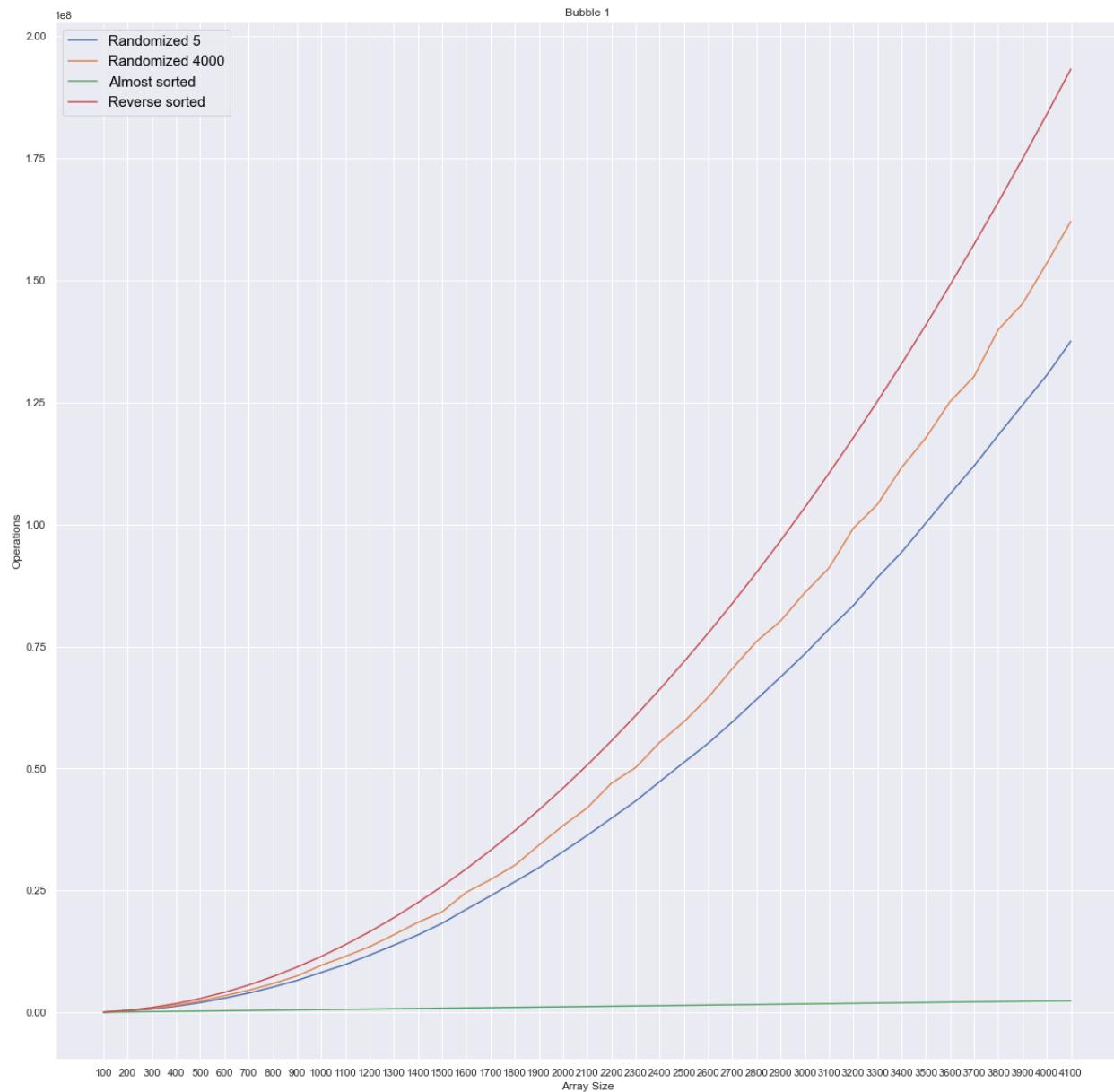
По четырём типам массивов меньше всего операций производится при сортировках подсчётом, Шелла, Шелла-Циура. На случайных значениях все виды пузырька проигрывают, и лишь на почти отсортированном массиве версии с оптимизациями 1 и 1+2 показывают не очень большое кол-во операций.

## Пузырьком



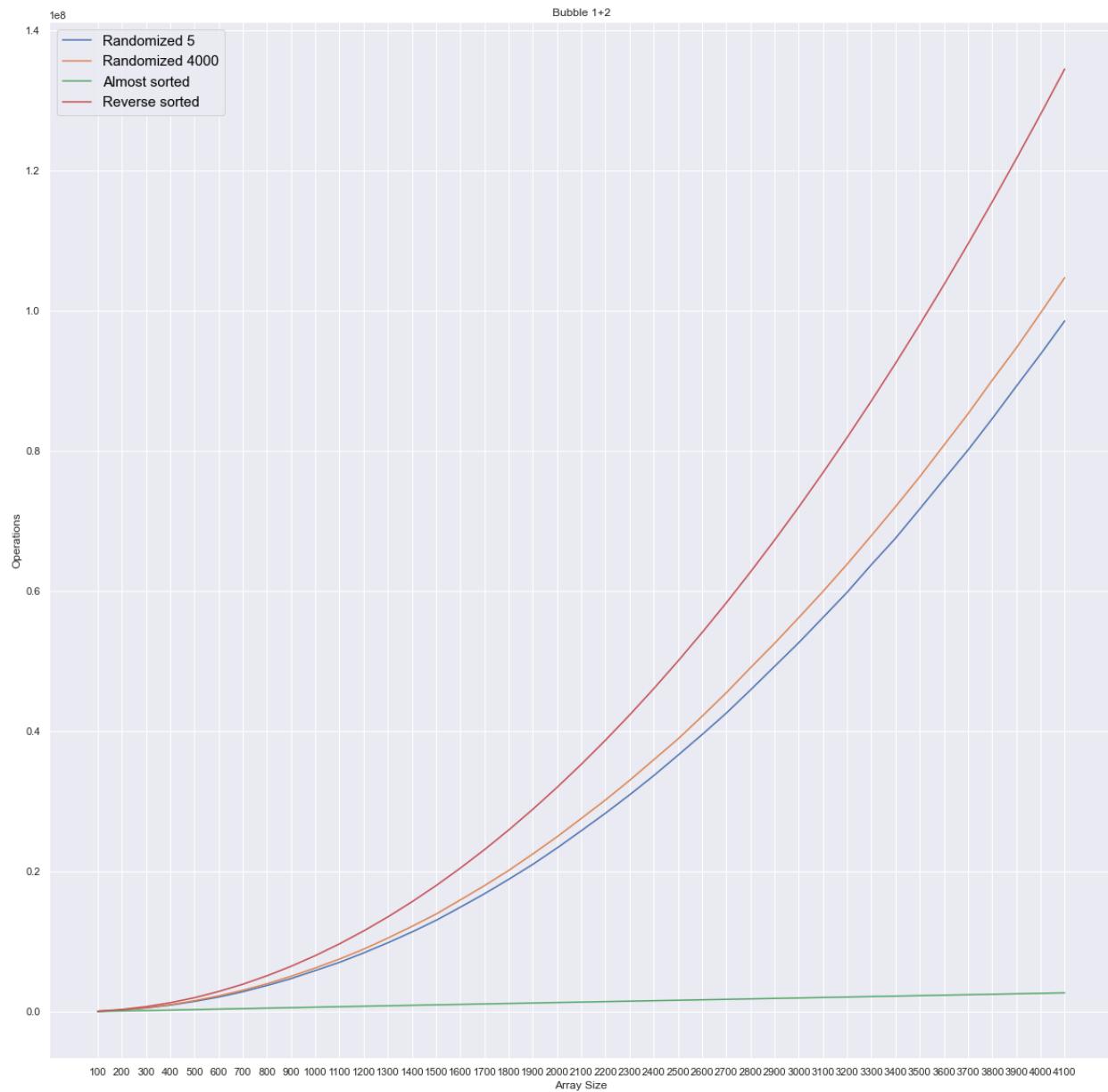
Как и ожидалось, худшим образом проявляется на обратно отсортированном массиве, в то время как на почти отсортированном меньше всего приходится делать перестановок.

## Пузырьком с оптимизацией Айверсона 1



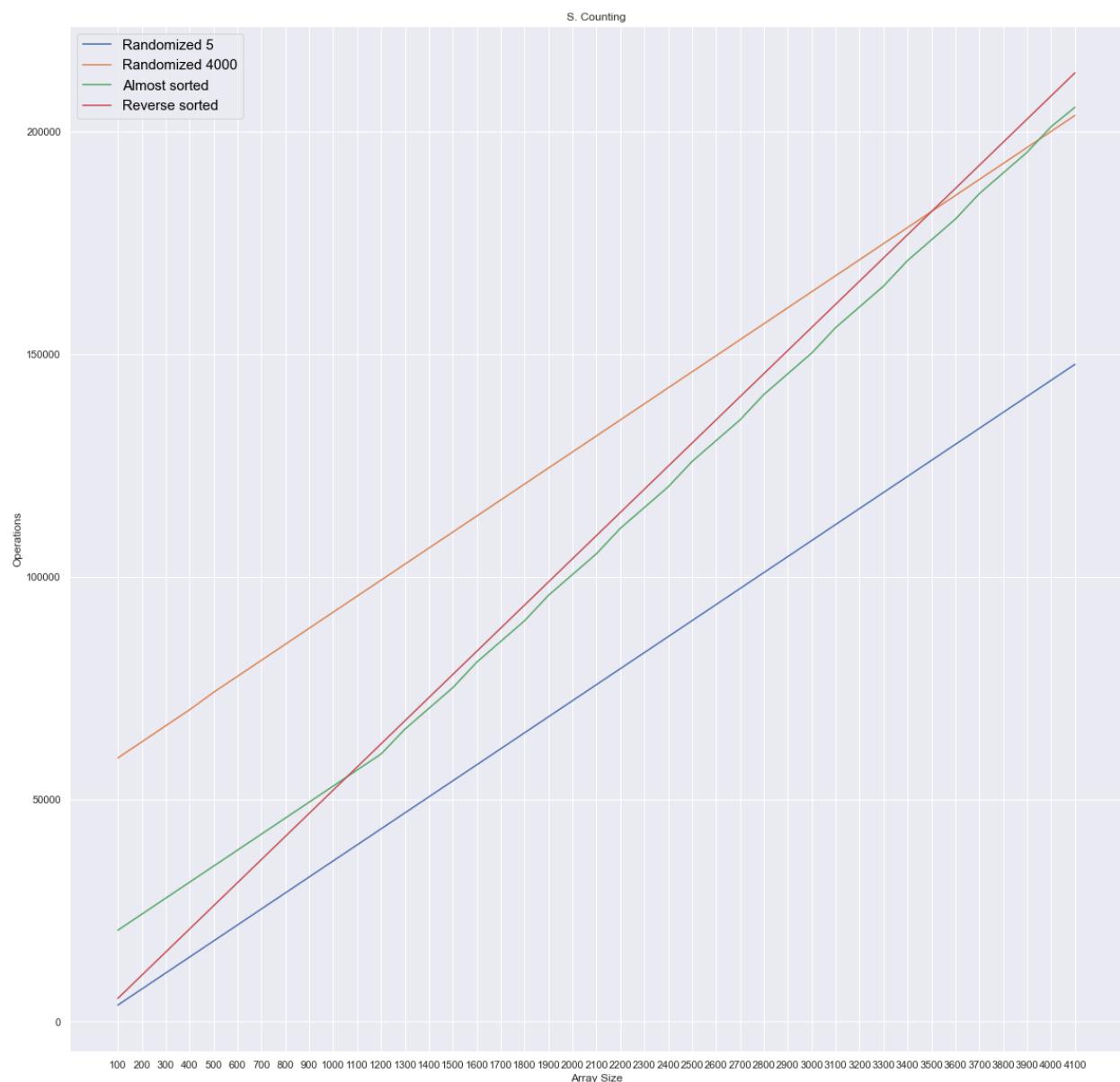
Всё аналогично предыдущему результату, однако у всех графиков кол-во операций намного меньше, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Пузырьком с оптимизацией Айверсона 1+2



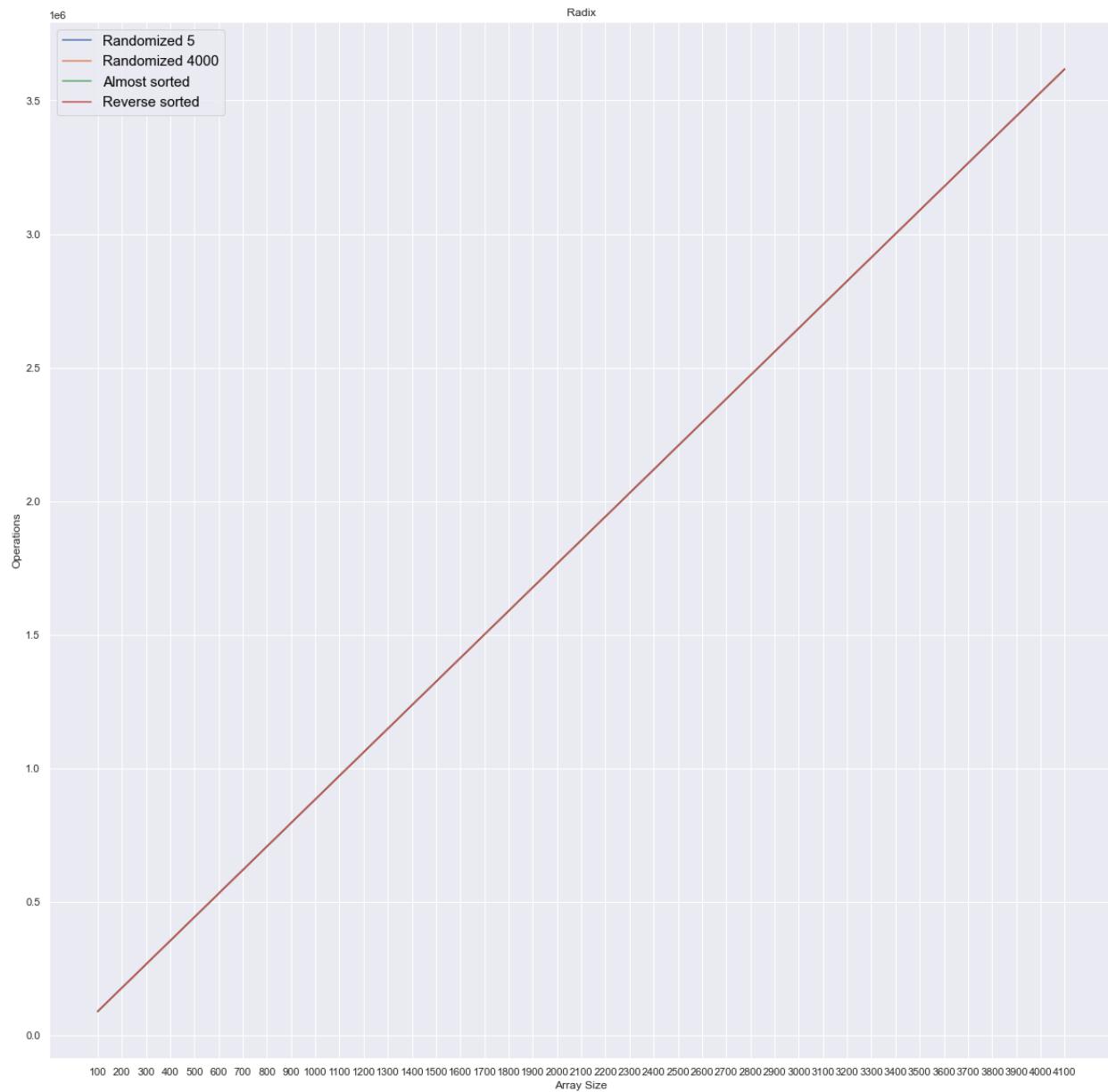
Всё аналогично предыдущему результату, однако у всех графиков кол-во операций намного меньше, особенно у почти отсортированного, что тоже довольно-таки очевидно.

## Подсчётом



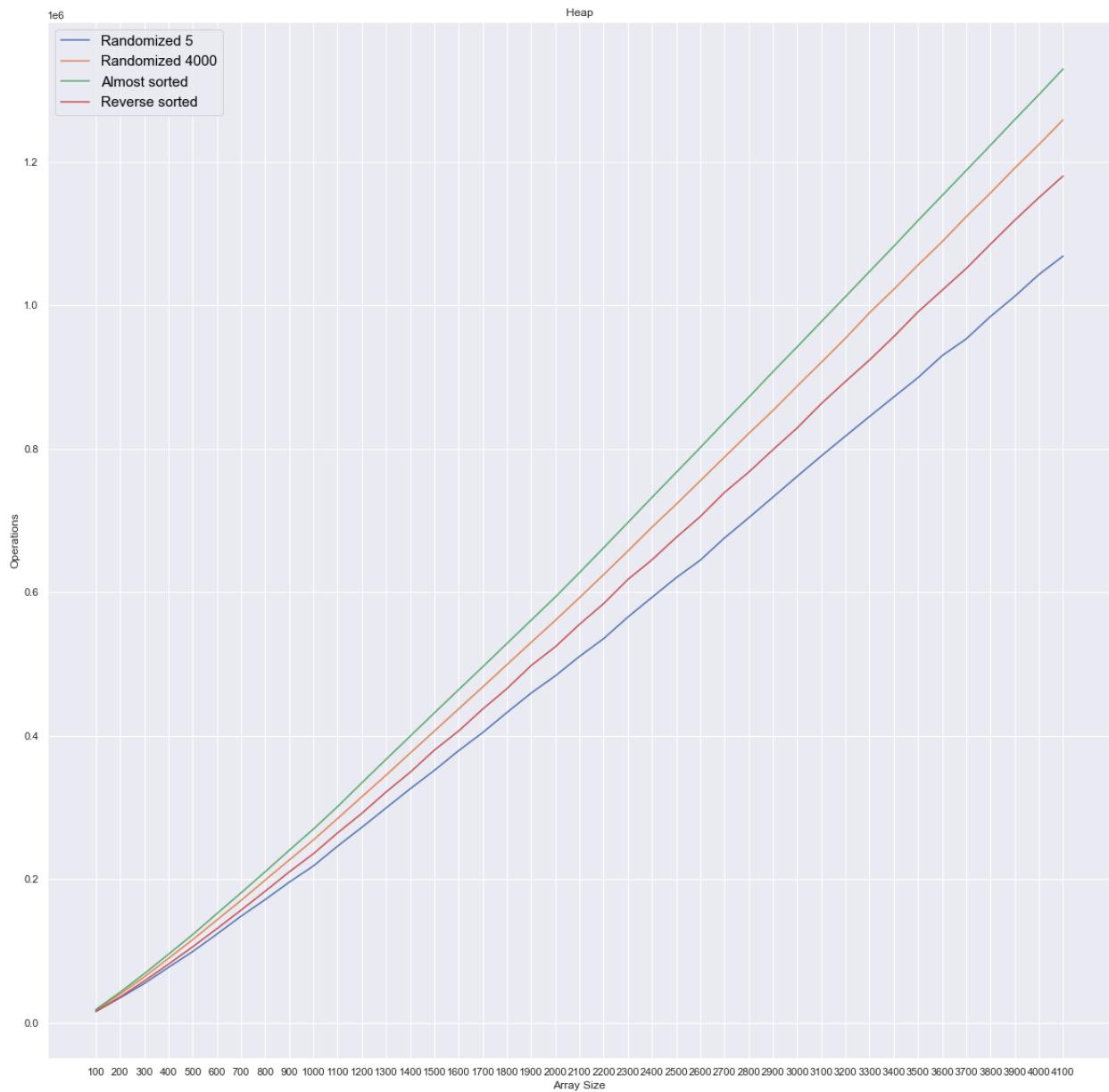
Прямо и обратно сортированные массивы с увеличением размера начинают совершать всё больше и больше операций, что, скорее всего, обусловлено тем, что при их генерации диапазон равен размеру, и следовательно количество операций возрастает.

## Цифровая



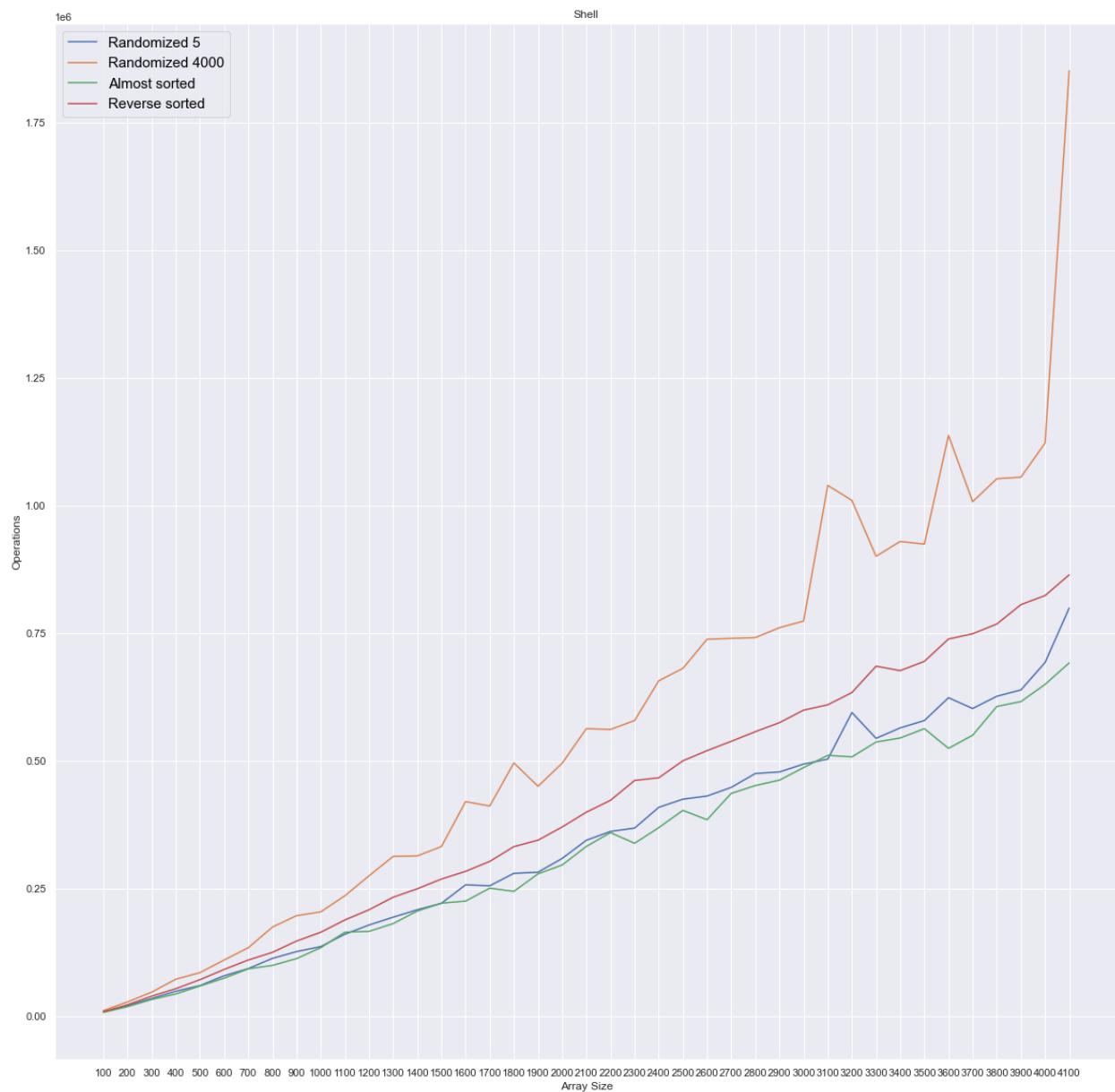
Видим, что цифровая сортировка работает линейно и одинаково для всех типов массивов.

## Пирамидальная



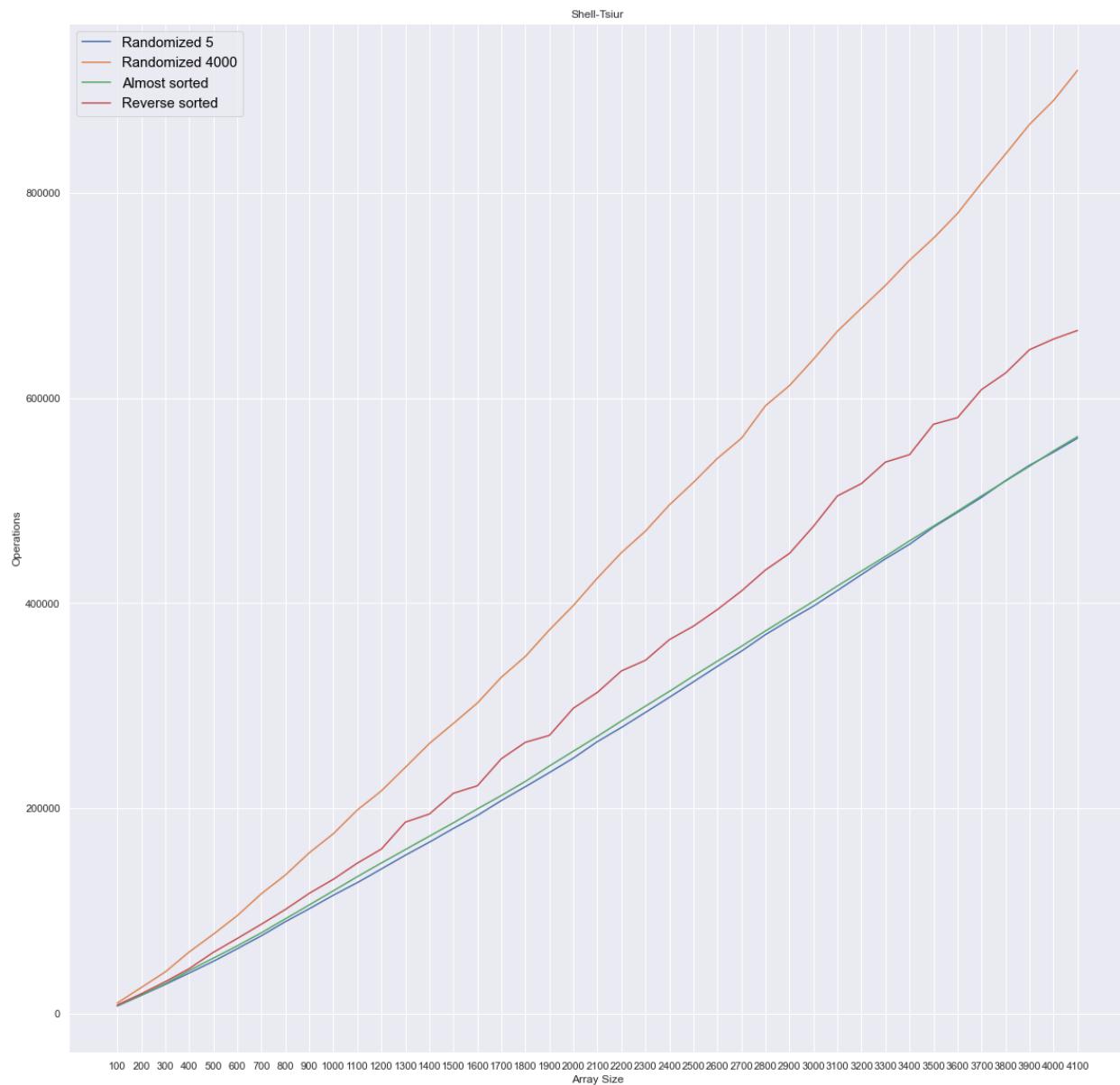
С увеличением размера массива графики больше отдаляются друг от друга. Меньше всего операций требуется рандомизированному массиву 5, а больше всего — почти отсортированному.

## Шелла



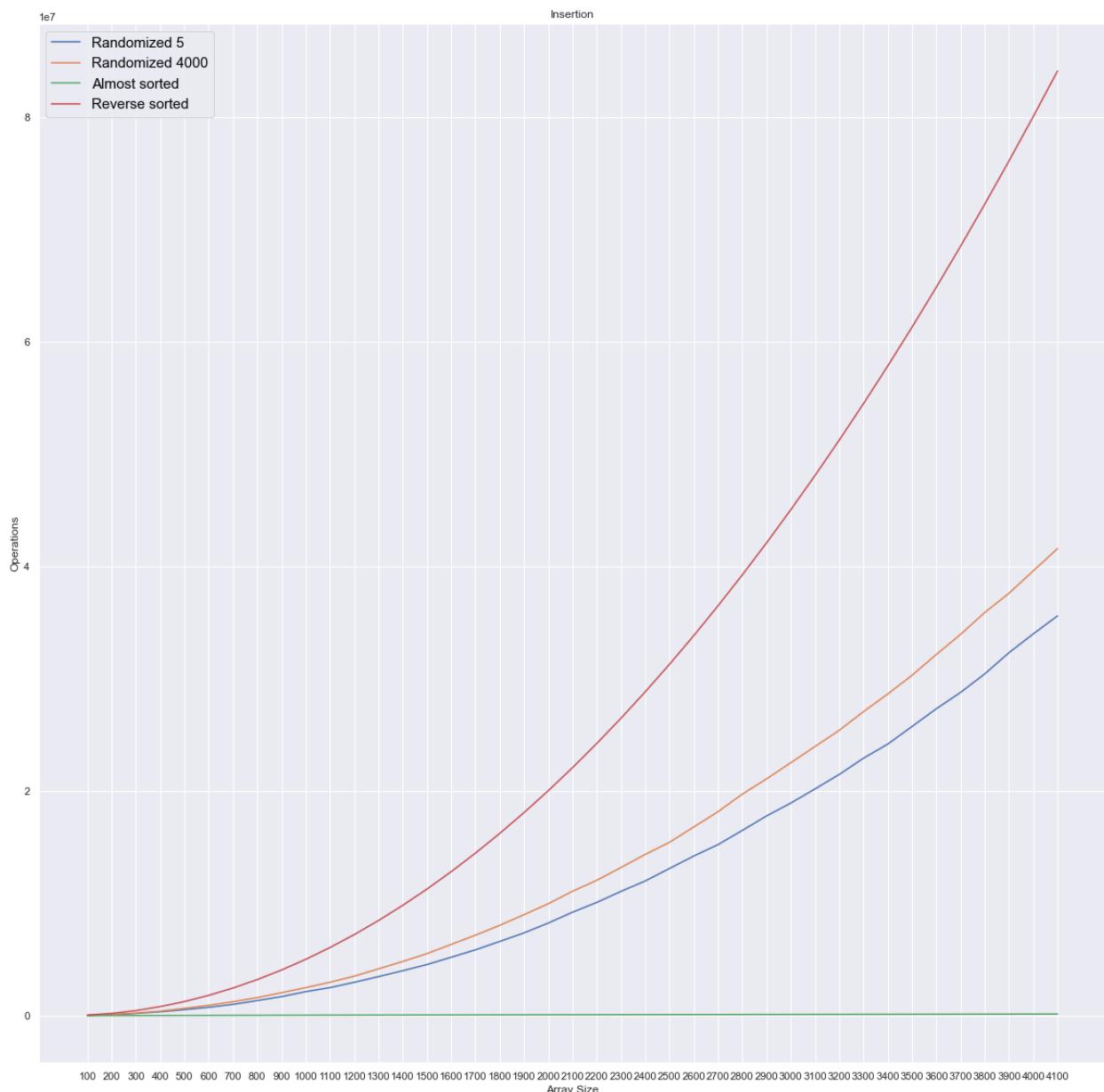
Больше всего операций производится на случайном массиве с диапазоном [1;4000], поскольку слишком часто приходится производить перестановки элементов, меньше всего — на почти отсортированном.

## Шелла-Циура



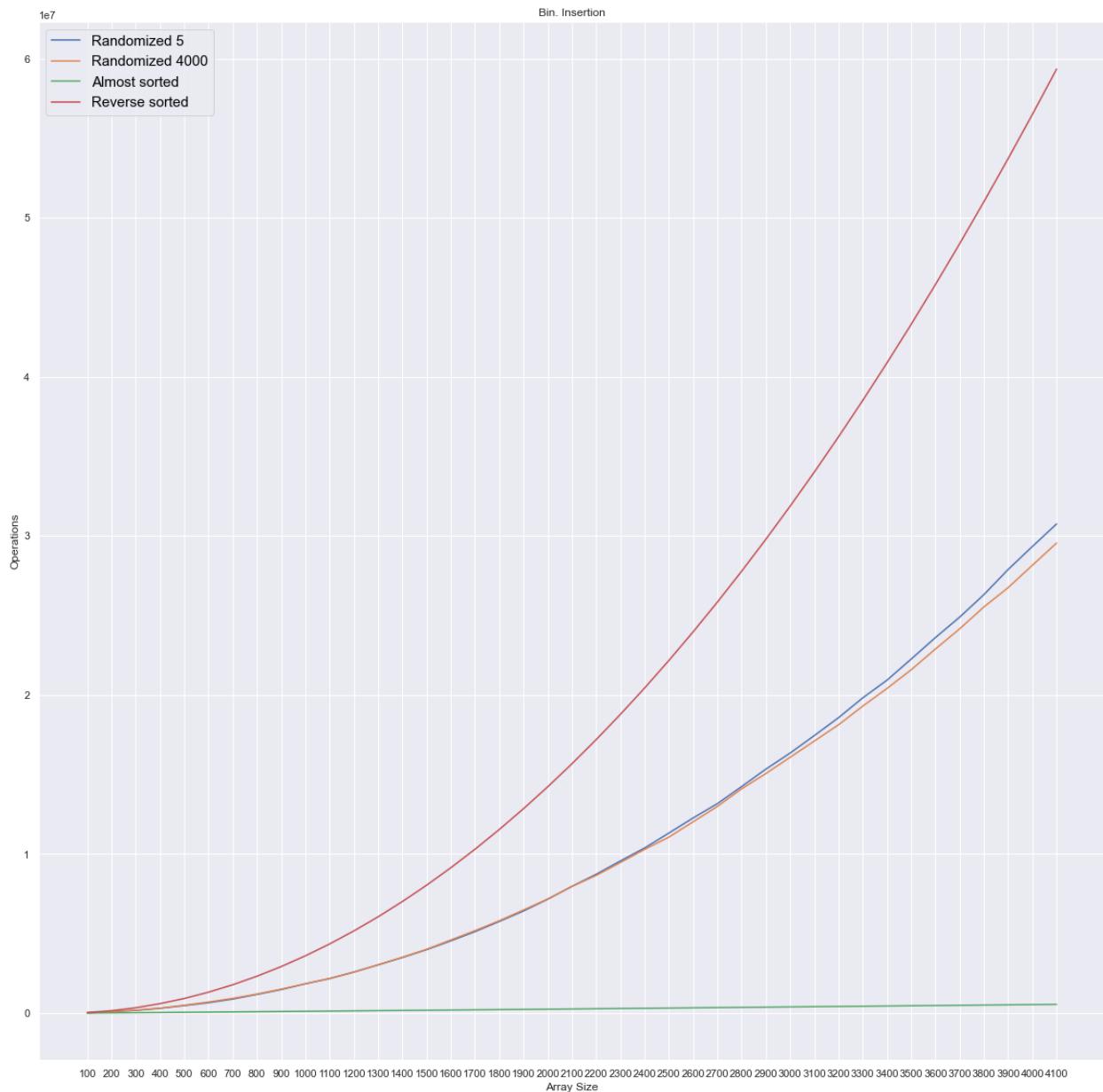
В отличии от предыдущей ситуации, здесь в целом сократилось кол-во операций у каждого типа массива, при этом синий и зелёный графики стали заметно меньше колебаться при увеличении размера массива.

## Вставками



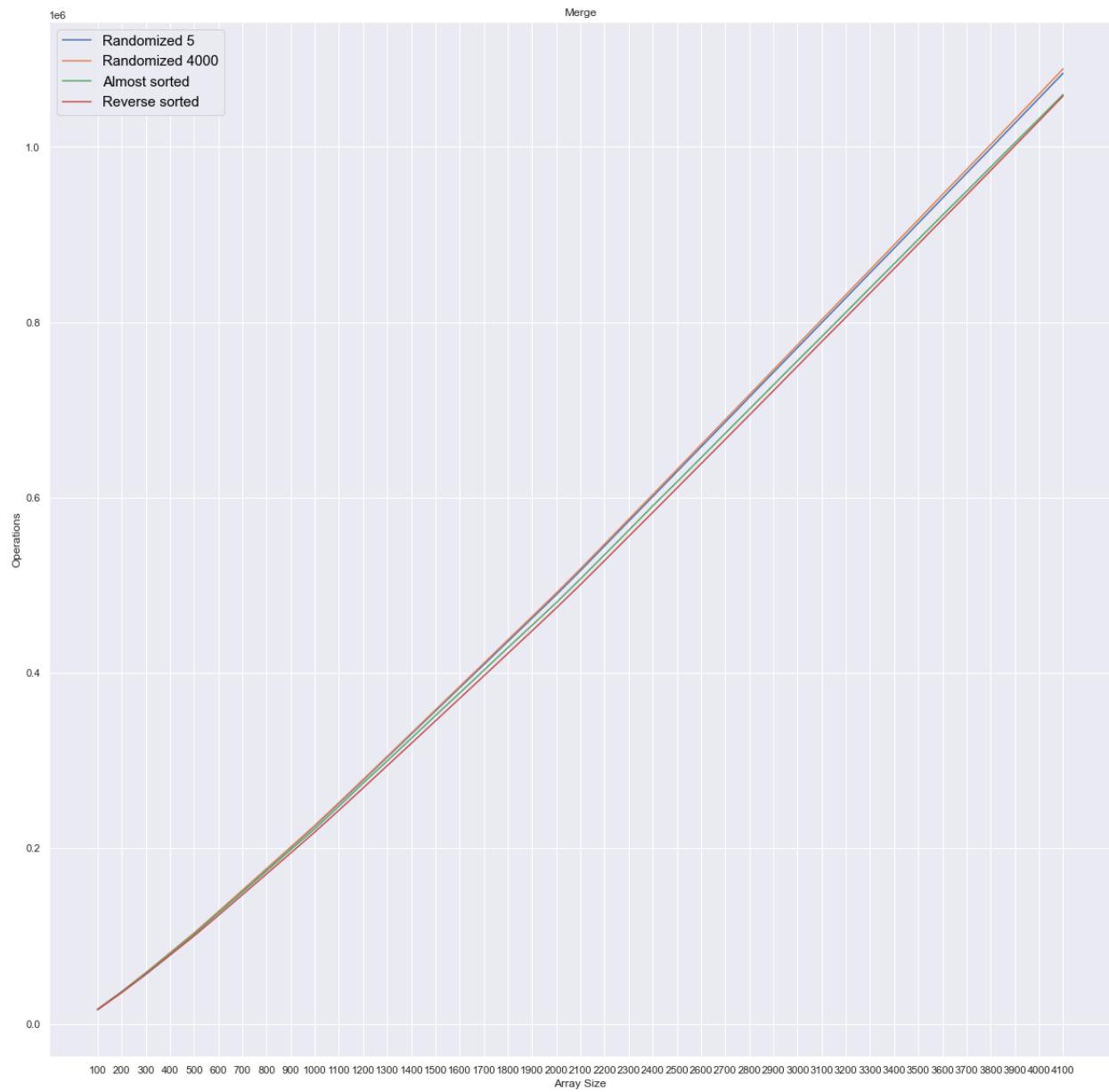
Как и ожидалось, очень много операций совершается при обратно отсортированном массиве, поскольку каждый раз приходится проходить весь отрезок до начала, прежде чем вставить элемент на нужную позицию (`array[0]`). Лучше всего показала себя при работе с почти отсортированным, поскольку в большинстве случаев элемент не приходится вставлять в другую позицию уже отсортированного подотрезка.

## **Бинарными вставками**



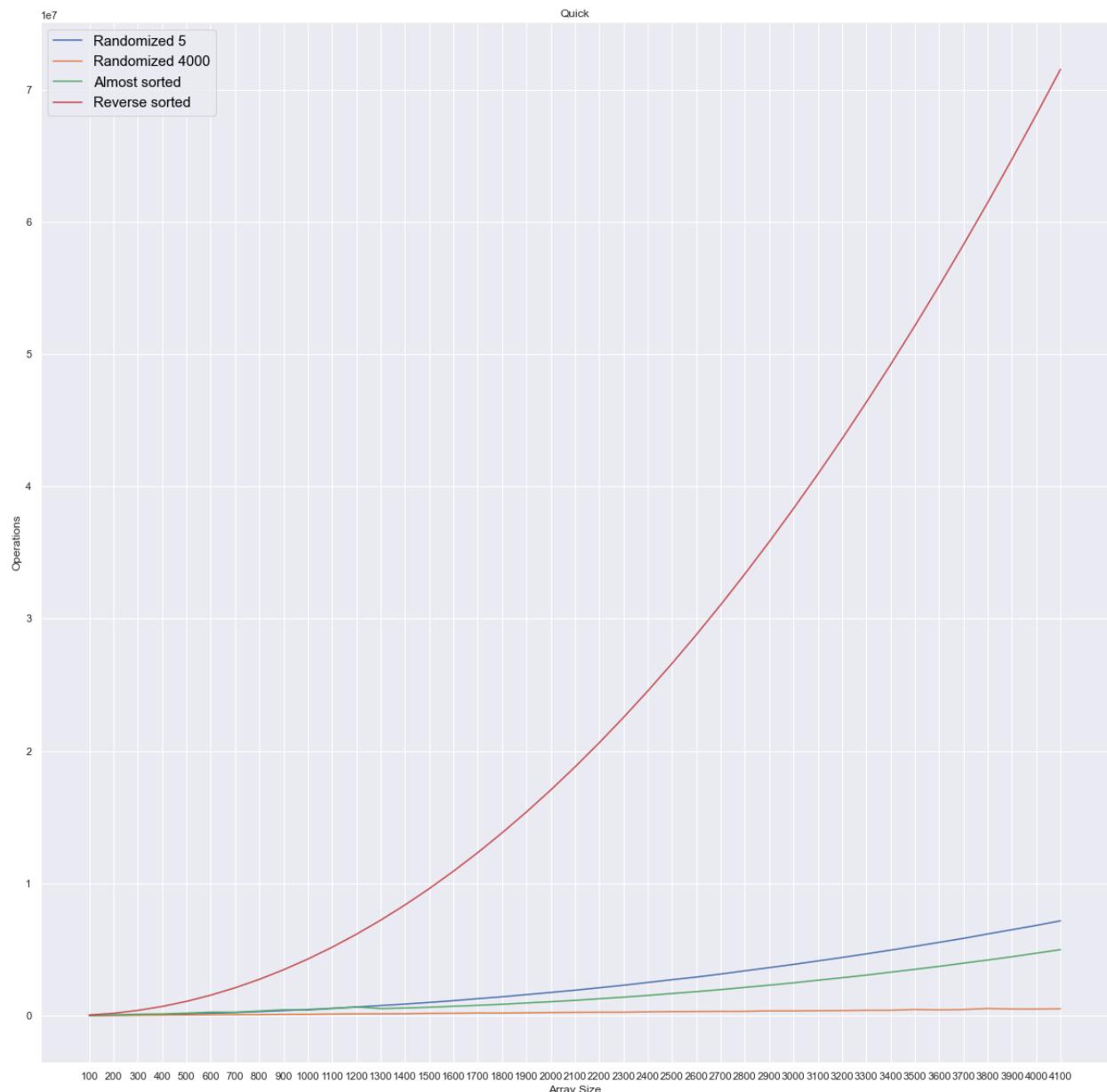
Количество операций всех массивов сократилось, а так наблюдается ситуация, аналогичная предыдущему случаю.

## **Слиянием**



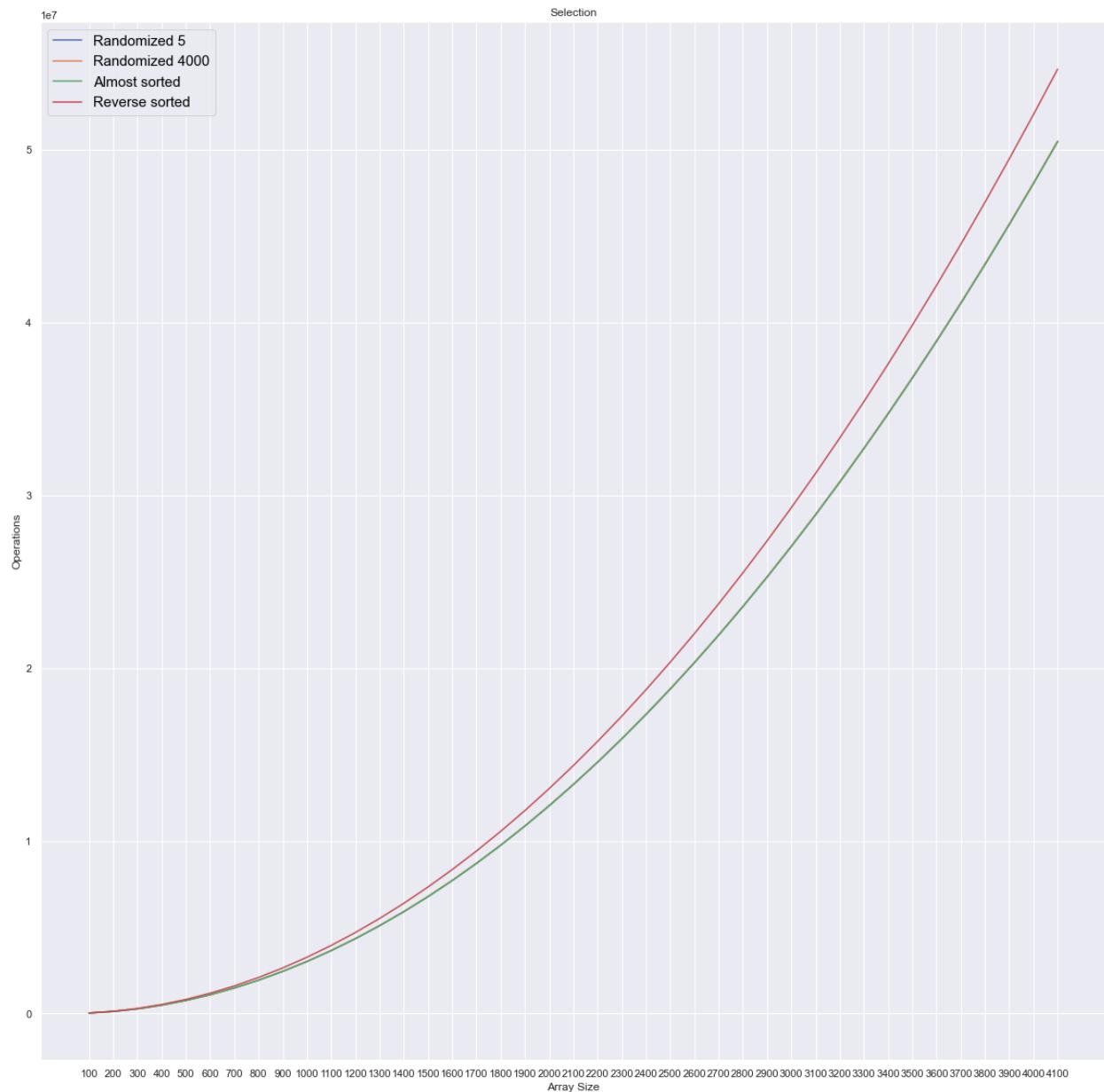
Результаты в целом похожие, немного меньше операций на максимально упорядоченных массивах (по возрастанию и убыванию). Графики незначительно расходятся с увеличением размера массивов.

## Быстрая



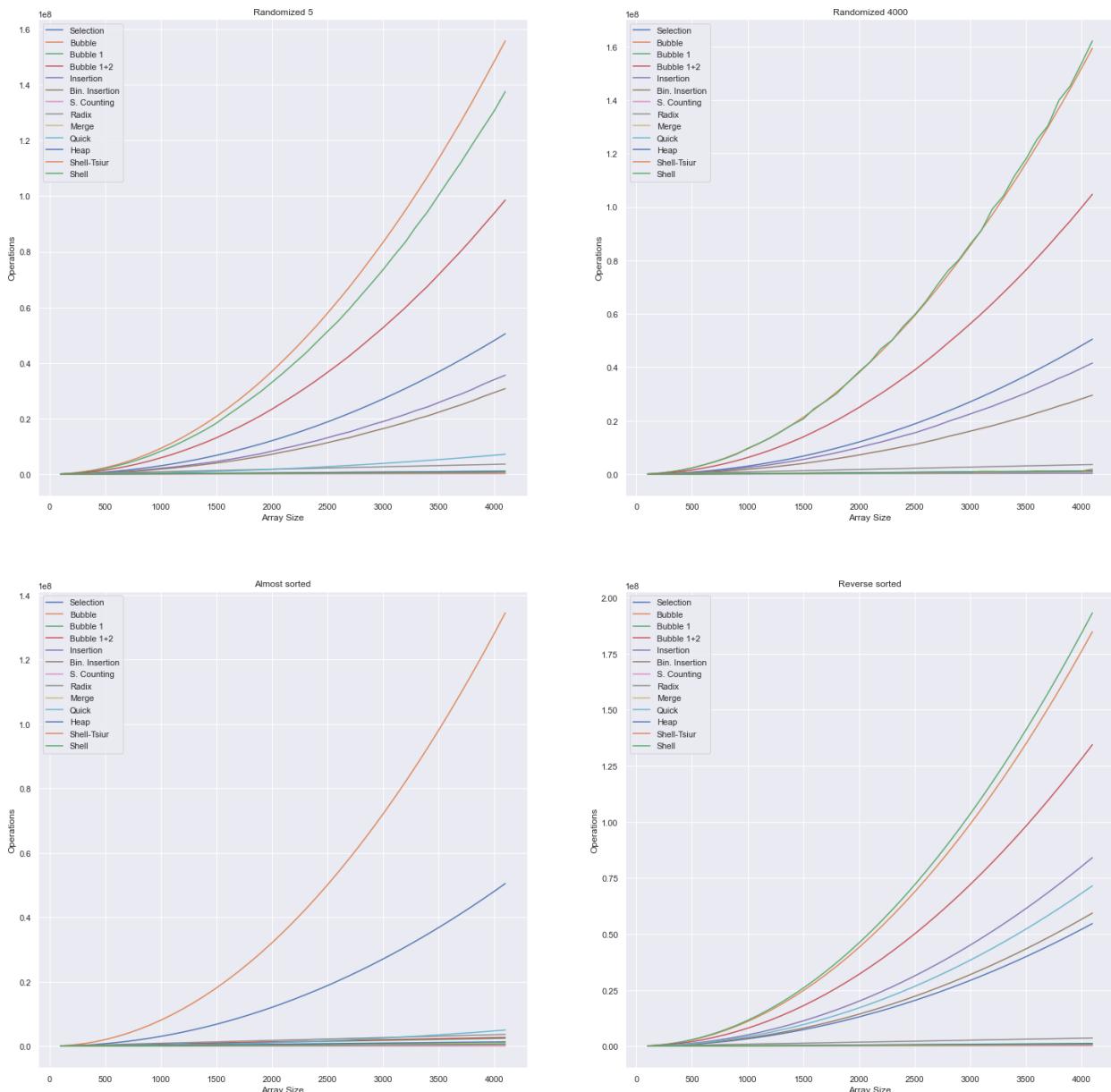
Очень много операций на обратно отсортированном массиве, поскольку за pivot берётся первый элемент подотрезка (т. е. максимальный для текущего подотрезка).

## Выбором



Почти одинаково работает на всём диапазоне размера массива, кроме обратно отсортированного — операций производится больше. Поскольку графики накладываются друг на друга, пришлось смотреть значения в таблице excel. Исходя из таблицы, графики Randomized 5 вместе с Randomized 4000, и Almost sorted вместе с Reverse sorted накладываются друг на друга.

# Сравнение сортировок



По четырём типам массивов меньше всего операций производится при сортировках подсчётом, цифровой, слиянием, Шелла, Шелла-Циура. На случайных значениях все виды пузырька проигрывают, и лишь на почти отсортированном массиве версии с оптимизациями 1 и 1+2 показывают малое число операций.

## Итог

Несмотря на 100 измерений для каждого из показаний времени при отключенном интернете и снятых ненужных задач с компьютера, время от времени мы всё равно встречали выбросы в показаниях. Они могут быть обусловлены большой нагрузкой на систему; IDE, в которой собирается и запускается программа; также отсутствием усреднений результатов при разных сгенерированных массивах (то есть для каждого из 4-ёх типов массивов сделать условно 10-30 генераций и усреднить значения).